Interface d’accès à *Code_Aster* : astk

Résumé :

Ce document présente astk (contraction de *Aster* et Tk, prononcer « astek »), l’interface permettant d’organiser ses calculs *Code_Aster*.
On décrit la logique de fonctionnement d’astk, l’interface graphique elle-même, bsf qui permet d’explorer les fichiers sur les différents serveurs, puis on explique comment l’utilisateur peut configurer astk en fonction de ses besoins.

Pour un accès rapide à *Code_Aster*, on montre à partir d'exemples comment lancer une étude, produire une surcharge, lancer une liste de cas-tests et comment faire une étude paramétrique.

Pour les personnes ayant accès au serveur de calcul *Code_Aster* du réseau EDF, on explique comment émettre une fiche d’anomalie, accéder aux fiches de retour d’expérience (REX), et, pour les développeurs, aux outils de l’AGLA (atelier de génie logiciel *Aster*).

Enfin, dans une seconde partie, on présente as_run qui offre de nombreuses possibilités depuis le lancement d'un calcul Aster jusqu'aux fonctionnalités avancées pour les développeurs.

Sont décrites ici les fonctionnalités d’astk version 1.8.3 qui gère toutes les versions supportées de *Code_Aster*.

Table des matières

1 Présentation et notions générales

La mise à disposition d'un outil comme *Code_Aster* qui fonctionne sur de multiples plates-formes nécessite une interface graphique pour simplifier le travail de l'utilisateur.
Par ailleurs, l'évolution des moyens de calcul centralisés et la diffusion en libre de *Code_Aster* ont nécessité le développement d'un produit portable dont l'architecture puisse s'adapter aux différentes configurations informatiques.

astk est l'interface graphique qui permet d'organiser ses calculs *Aster* : préparer ses données, organiser les fichiers, accéder aux outils de pré et post-traitement, lancer et suivre l'évolution des calculs. astk permet également de choisir la version de *Code_Aster* à utiliser parmi celles disponibles (STA, NEW…).

Le nom « astk », prononcer « astek », vient de la contraction de *Aster* et Tk qui est la librairie graphique utilisée.

1.1 Démarrage

Pour lancer l'interface, il suffit de taper dans un terminal : `astk`

En général, la commande a été placée dans le chemin par défaut des utilisateurs, l'interface s'ouvre avec une configuration par défaut (voir [*§ 4] Configuration*).
Si la commande n'est pas trouvée, il faut lancer : `[aster_root]/bin/astk` où `[aster_root]` est le répertoire d'installation de *Code_Aster* (contactez éventuellement votre administrateur).

1.2 Fonctionnalités

***ETUDE** :* astk permet de lancer un calcul *Aster* sur la machine locale (par exemple dans le cadre d'une utilisation sur ordinateur personnel), sur un serveur de calcul départemental ou bien sur le serveur de calcul *Aster* de EDF-R&D (accès restreint à EDF et ses prestataires d'études).

***SURCHARGE** :* La diffusion du code source de *Code_Aster* autorise chacun à tester ses propres développements. astk permet de "surcharger" le code, c'est-à-dire ajouter ou modifier des fichiers sources, de créer une version particulière et l'utiliser sur des cas-tests ou pour des études. On peut ainsi créer et utiliser de nouveaux exécutables, catalogues de commandes ou d'éléments, et modules python.

***TESTS** :* Il est souvent judicieux de tester que ses propres développements n'impactent pas le code par ailleurs ; astk permet de lancer facilement une liste de cas-tests avec une version personnelle.

***AGLA** :* Aux développeurs de la version de référence EDF, astk offre l'accès à l'atelier de génie logiciel *Aster*, AGLA, qui permet d'organiser le développement collaboratif de la version de développement (éviter les conflits, assurer la non régression…).

***REX** :* Faire vivre le code passe par la prise en compte des remarques, besoins, et avis des utilisateurs. Des fiches de retour d'expérience peuvent être émises et consultées par les utilisateurs depuis l'interface, les développeurs peuvent y répondre (accès au serveur EDF requis). Cette fonctionnalité est réservée à EDF et à ses prestataires.

MULTI-MACHINES : Les fichiers nécessaires à ces différentes actions (fichiers de maillage, source, résultats…) peuvent être répartis sur différentes machines sur le réseau (déclarées dans astk), astk assurant le transfert et la compression/décompression.
Le passage d'un coupe-feu n'est pas proposé.

OUTILS : L'utilisateur peut lancer différents outils pré-définis et configurer ceux dont il a besoin (mailleur, outil de post-traitement, éditeur…).

BSF : Un navigateur est fourni (appelé `bsf`, boîte de sélection de fichiers), il permet de parcourir les systèmes de fichiers des machines distantes définies et d'effectuer des opérations courantes sur les fichiers : copie, suppression, changement de nom, impression, ou encore d'ouvrir une fenêtre de commandes sur ces machines.

ASJOB : On peut suivre les calculs, leur état (notamment dans le cas de lancement en batch) depuis la fenêtre « Suivi des jobs », aussi appelée `asjob`.

1.3 Mode de fonctionnement

L'architecture client/serveur autorise une séparation nette entre l'interface (client) et les outils utilisés pour accéder au code (services). Le point d'entrée du serveur est `as_run` : il permet d'utiliser l'ensemble des scripts de l'AGLA pour gérer la version de référence en batch et/ou d'accéder à toutes les versions disponibles sur un réseau ou en local.
Le protocole de communication entre les différentes machines est rsh ou ssh pour les commandes shell et rcp ou scp pour la copie de fichiers.

Exemple : Le client (c'est-à-dire l'interface lancée par la commande `astk` ou `codeaster-gui`) demande le lancement d'un calcul sur un serveur de calcul.
L'interface exécute un service sur le serveur en question (commande rsh ou ssh) en lui indiquant les paramètres nécessaires au calcul (contenus dans un profil, voir [§ 1.5]). Le service prend ensuite en charge les opérations nécessaires : rapatrier les fichiers nécessaires au calcul des différentes machines vers le serveur de calcul, lancement du calcul, retour des fichiers résultats…

Pour les puristes : `as_run` n'est pas véritablement un serveur dans les faits, il n'y a pas de démon à l'écoute sur un port particulier. Il est démarré à la demande.

1.4 Serveurs astk

Pour astk, un serveur est :

- soit un serveur de calcul *Aster* : une machine sur laquelle on peut trouver `as_run`, c'est-à-dire l'ensemble des services qui permettent d'accéder au code ; on pourra lancer des calculs via ce serveur et utiliser des fichiers sur ce serveur pour un calcul,
- soit un serveur de fichiers : on pourra simplement utiliser des fichiers sur ce serveur lors d'un calcul.

Le serveur appelé "Local" est en fait un serveur de fichiers (seules les informations de connexion sont nécessaires pour un serveur de fichiers, or on les connaît facilement sur la machine locale.
Si l'on souhaite lancer des calculs sur la machine locale (sur laquelle sont installées une version d'*Aster* et la partie serveur d'astk), il faut aussi déclarer cette machine comme un serveur de calcul (Local et Machine dans l'exemple suivant et §2.1.2).

Dans ce document, on appellera « serveur de référence » la machine sur laquelle est gérée la version de développement de *Code_Aster* en interne EDF. Certaines fonctionnalités sont accessibles uniquement si l'utilisateur a un accès à cette machine.

Exemple de configuration : (voir [§ 2.1.2])

astk est utilisé sur la machine de nom `mach00`, sur laquelle est installé *Code_Aster*. On a accès à un serveur de fichiers départemental `file01`, les fichiers de ce serveur sont accessibles (montage nfs) depuis deux machines de calcul `comp02`, `comp03`. On a aussi accès à un cluster avec N nœuds de calcul dont la machine frontale est `front04`.

On a alors :

- « Local » (label réservé) : on l'a toujours pour explorer les fichiers qui se trouvent sur `mach00` ;
- « Machine » (label quelconque, la procédure d'installation fixe ce nom à la valeur retournée par la commande `` `uname -n` ``) : dont l'adresse IP est celle de `mach00` qui est le serveur de calcul (services as_run installés) de la machine locale ;
- « Depart » (label quelconque) : dont l'adresse IP est celle de `file01`, qui permet d'explorer les fichiers hébergés par le serveur de fichier, configuré comme un serveur de calcul Aster (services as_run installés) ayant `comp02` et `comp03` vus comme des nœuds de calcul (`comp0i` peut être la même machine que `file01`) ;
- « Cluster » (label quelconque) : dont l'adresse IP est celle de `front04` qui est un serveur de calcul (services as_run installés) ayant N nœuds, la seule machine accessible étant `front04`.

La différence entre « Depart » et « Cluster » est qu'en général les stations de calcul sont accessibles directement (on pourrait donc soumettre un calcul en interactif sur l'une ou l'autre), alors que pour un cluster, les utilisateurs ne voient en général que la machine frontale, la répartition étant faite sur les nœuds par un séquenceur de travaux batch.

On peut aussi bien avoir plusieurs serveurs de calcul « Depart » ou « Cluster » que n'avoir uniquement le poste « Local+Machine ».

1.5 Profil astk

Un profil astk est un fichier qui contient toutes les informations relatives à une étude, une surcharge, etc. : l'emplacement des fichiers de l'étude, en donnée, en résultat, le type associé à chaque fichier, les paramètres de soumission du calcul (mémoire, temps, machine de calcul, batch/interactif…), en cas de surcharge, l'emplacement des fichiers sources, de l'exécutable, des catalogues produits…

Le profil contient également des paramètres sur l'interface elle-même pour reprendre une étude exactement dans le même état où on l'avait laissée.
Le profil est enregistré sur demande de l'utilisateur (menu *Fichier/Enregistrer* ou *Enregistrer sous…*) et à chaque lancement d'un calcul (il s'agit du fichier dont l'extension est, par convention, `.astk`).

On parlera plus loin du fichier « export » (extension `.export`) qui est une représentation simplifiée du profil astk : il ne contient que les informations sur les fichiers et les paramètres nécessaires au calcul (et aucune information sur l'interface graphique).

2 Description de l'interface graphique

L'interface graphique (IHM) se décompose en 4 parties :

1. Une barre de menu
2. Les onglets
3. Les paramètres de soumission
4. La barre d'état
5. Zone des arguments Aster

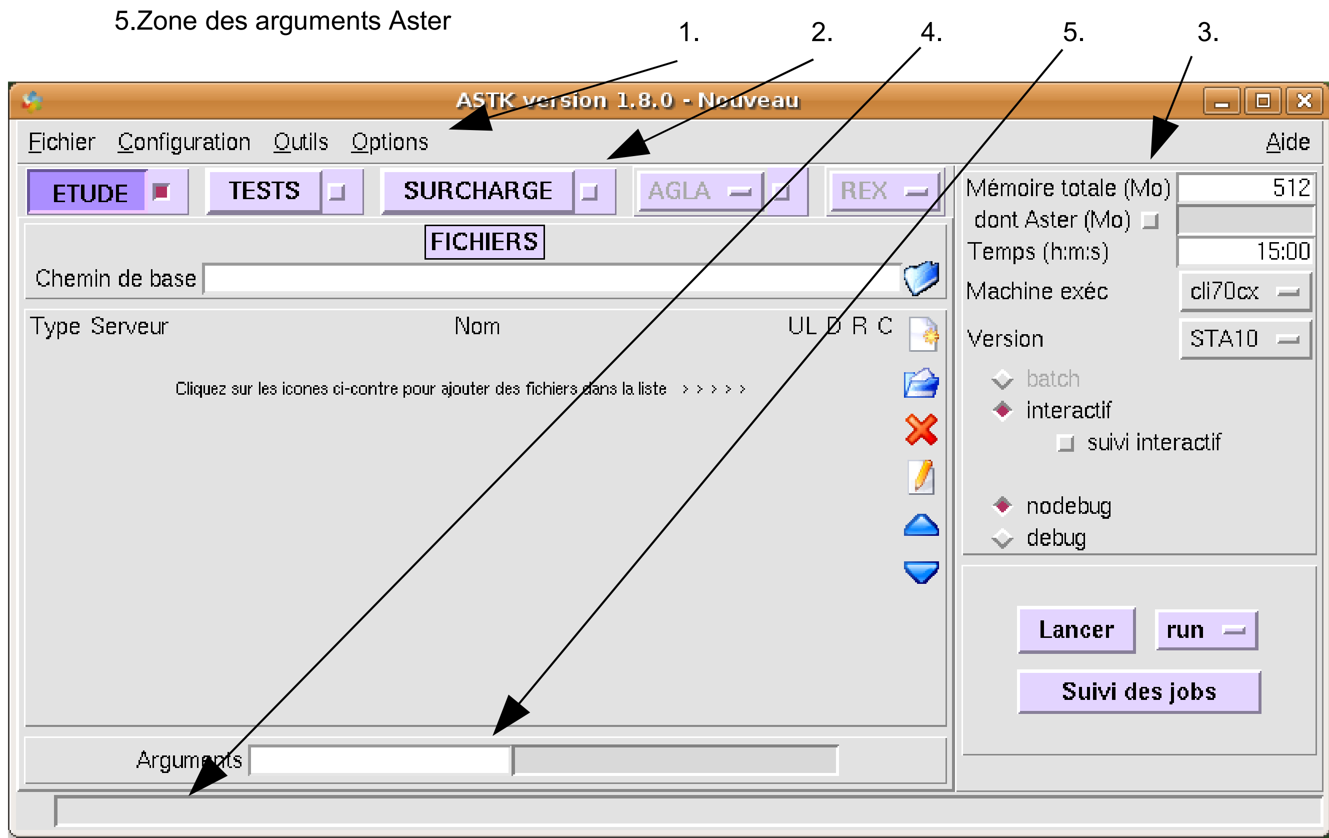


Figure 2-1: Fenêtre principale

2.1 Barre de menu

Le menu « Aide » permet également d'accéder à la description des menus.

2.1.1 Menu Fichier

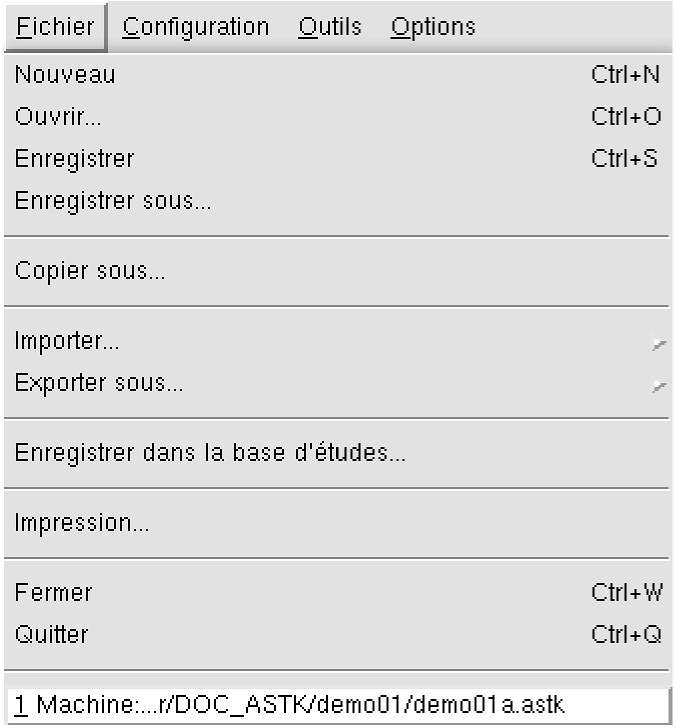


Figure 2.1.1-1: Menu Fichier

- **Nouveau** : Permet de créer un nouveau profil.
- **Ouvrir** : Ouvre un profil créé auparavant par `astk`.
- **Enregistrer** : Sauvegarde le profil courant (ceci est fait automatiquement lors du lancement d'un calcul).
- **Enregistrer sous** : Sauvegarde le profil courant en donnant le choix du nom du profil et de son emplacement.
- **Copier sous** : Copie le profil courant dans un autre répertoire et propose de copier les fichiers (en données et/ou résultats) qu'il référence.
- **Importer** :
  - `.export astk` : Relit un fichier `.export` d'une exécution précédente (cela peut être le fichier `*.pNNN` du répertoire `$HOME/flasheur`).
  - `.export astk (mode ajout)` : contrairement à l'import classique qui part d'un profil vierge, les fichiers et répertoires présents dans le fichier .export sont ajoutés au profil actuel. Les paramètres et arguments sont ignorés. Cela permet par exemple d'importer une étude dans un profil de surcharge.
  - `.export d'une fiche REX` : Importe les fichiers attachés à une fiche du REX dont on donne le numéro.
  - `cas-test` : Importe les fichiers nécessaires au lancement d'un cas-test. Les fichiers sont pris sur la machine d'exécution sélectionnée.
  - `cas-test (mode ajout)` : idem + les fichiers sont ajoutés à ceux déjà présents dans le profil (y compris la surcharge).
- **Exporter sous** : Permet de produire le fichier `.export` du profil courant.

- **Enregistrer dans la base d'études** : Permet d'ajouter le calcul courant dans la base de données d'étude en tant qu'exécution d'une étude existante (uniquement avec un accès au serveur de référence).

- **Impression** : Configuration des commandes d'impression (très peu utilisé).
  L'utilisateur définit des commandes d'impression (`lpr, a2ps...`) qui seront utilisées pour imprimer les fichiers, et les files d'impression (imprimantes). La commande peut être exécutée sur un serveur distant.
  Au code @P de la ligne de commande sera substitué le nom de la file d'impression, le nom du fichier remplacera @F (voir [§ 2.1.3] pour les codes de champs).
  Exemple de commande d'impression :
  ```
  lpr -h -P @P @F
  ```
  ou
  ```
  /usr/bin/a2ps -P @P -2 --print-anyway=yes --sides=tumble @F
  ```

- **Fermer** : Fermer le profil courant

- **Quitter** : Termine l'application

- Les N derniers profils ouverts sont directement accessibles à partir du menu Fichier.

2.1.2 Menu Configuration

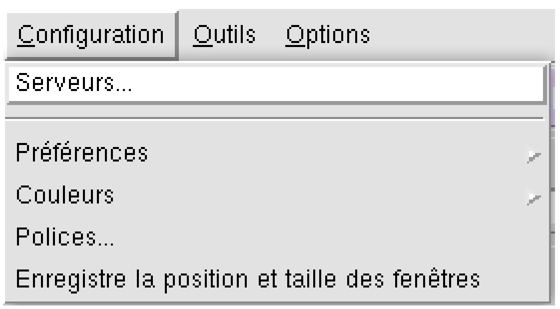


Figure 2.1.2-1: Menu Configuration

- **Serveurs** : Permet de modifier la configuration des serveurs : serveurs de calcul ou serveurs de fichier (voir Figure 2.1.2-2).
  Pour les serveurs de calcul, on choisit le mode de téléchargement de la configuration *Aster* : le serveur retourne la liste des versions disponibles, le mode d'exécution (batch et/ou interactif) et les limites associées.
  Il faut indiquer le 'login' à utiliser pour se connecter au serveur, et le répertoire où est installé *Code_Aster*.
  Pour les serveurs de fichier, choisir "aucun" comme mode de téléchargement de la configuration Aster.

- **Préférences** : Définit les préférences de l'utilisateur :

  - <u>Générales</u> (Voir Figure 2.1.2-3)
    Pour `astk` :
    Nom, prénom, email seront utilisés pour le lien vers l'outil de gestion du retour d'expérience (émission, consultation des fiches d'anomalies, d'évolution...).
    L'accès aux fonctions de l'atelier de génie logiciel *Aster* si l'utilisateur est reconnu en tant que développeur Aster sur le serveur de référence.
    Le chemin d'accès à l'éditeur et au terminal (fenêtre xterm) sur la machine locale, la version à sélectionner par défaut, le nombre de profils rémanents dans le menu Fichier, le niveau de message (debug).
    Pour le suivi des jobs :
    Le nombre de lignes affichées lors de la consultation d'un calcul en cours d'exécution (tail), et la fréquence d'actualisation automatique en minutes.
  - <u>Réseaux</u> (Voir Figure 2.1.2-4)

Le nom de domaine réseau de la machine et si le mode DHCP/VPN (adresse IP dynamique) est actif, possibilités de fixer la variable DISPLAY pour l'affichage des applications externes, les protocoles de communication utilisés pour les commandes shell (rsh ou ssh) et la copie de fichiers (rcp, scp ou rsync).
Si on travaille uniquement en local ou si toutes les machines sont dans le même domaine réseau, on peut laisser le nom de domaine vide et ignorer l'avertissement au démarrage. Dans le cas contraire, le nom de domaine est indispensable pour que les autres machines puissent contacter (copie de fichiers, affichage) la machine où est lancée astk.
Attention : utiliser l'option DISPLAY en connaissance de cause et uniquement si la valeur par défaut ne convient pas. Laisser le champ vide pour laisser astk déterminer seul le DISPLAY (en fonction de sa valeur au lancement d'astk, du nom de domaine...).

- **Couleurs** : `Classique`, `KDE3 Crystal`, `Nostalgique`, `Personnalisé`,...
- **Polices**
- **Enregistre** la position et taille des fenêtres

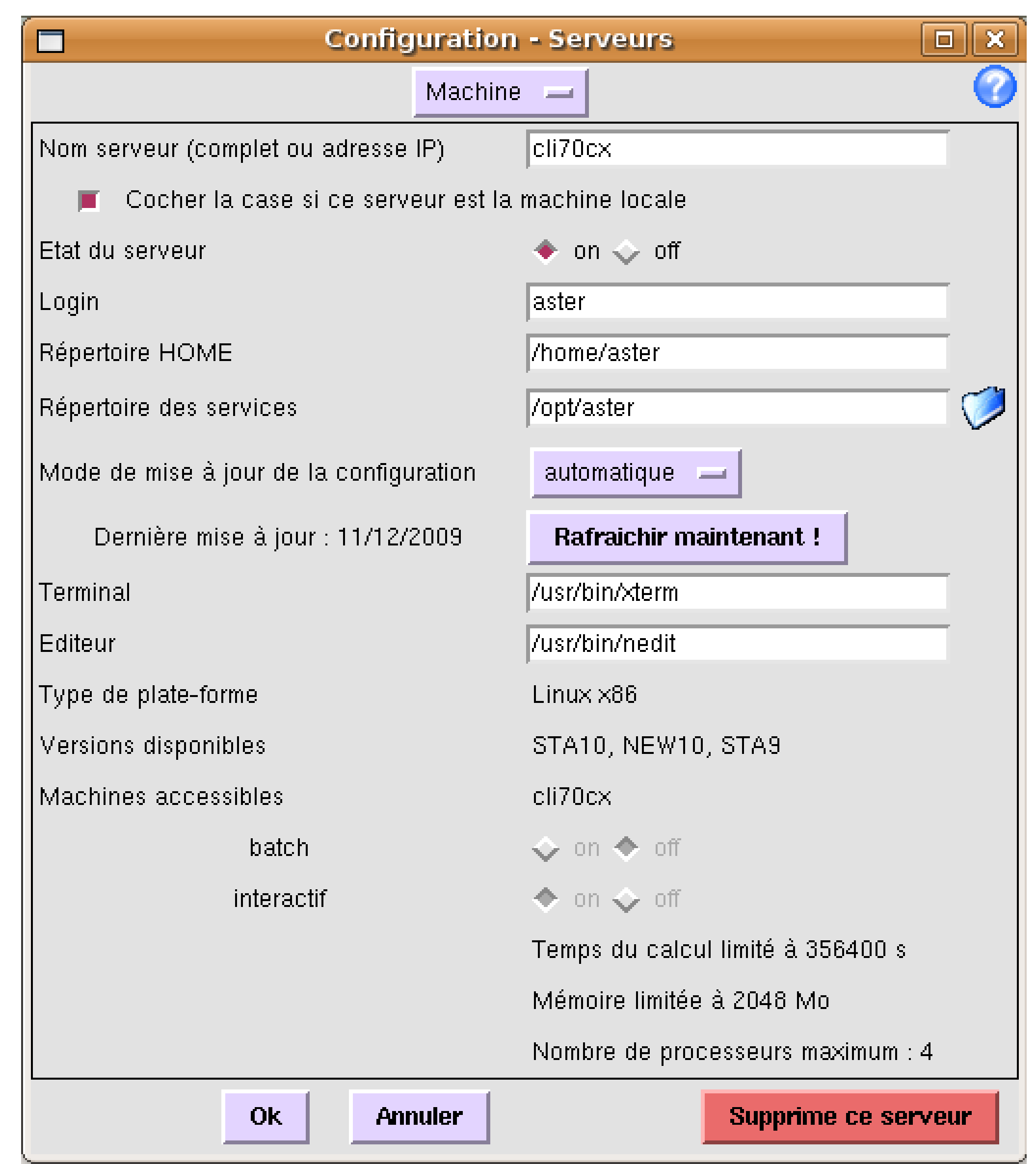


Figure 2.1.2-2: Menu Configuration/Serveurs

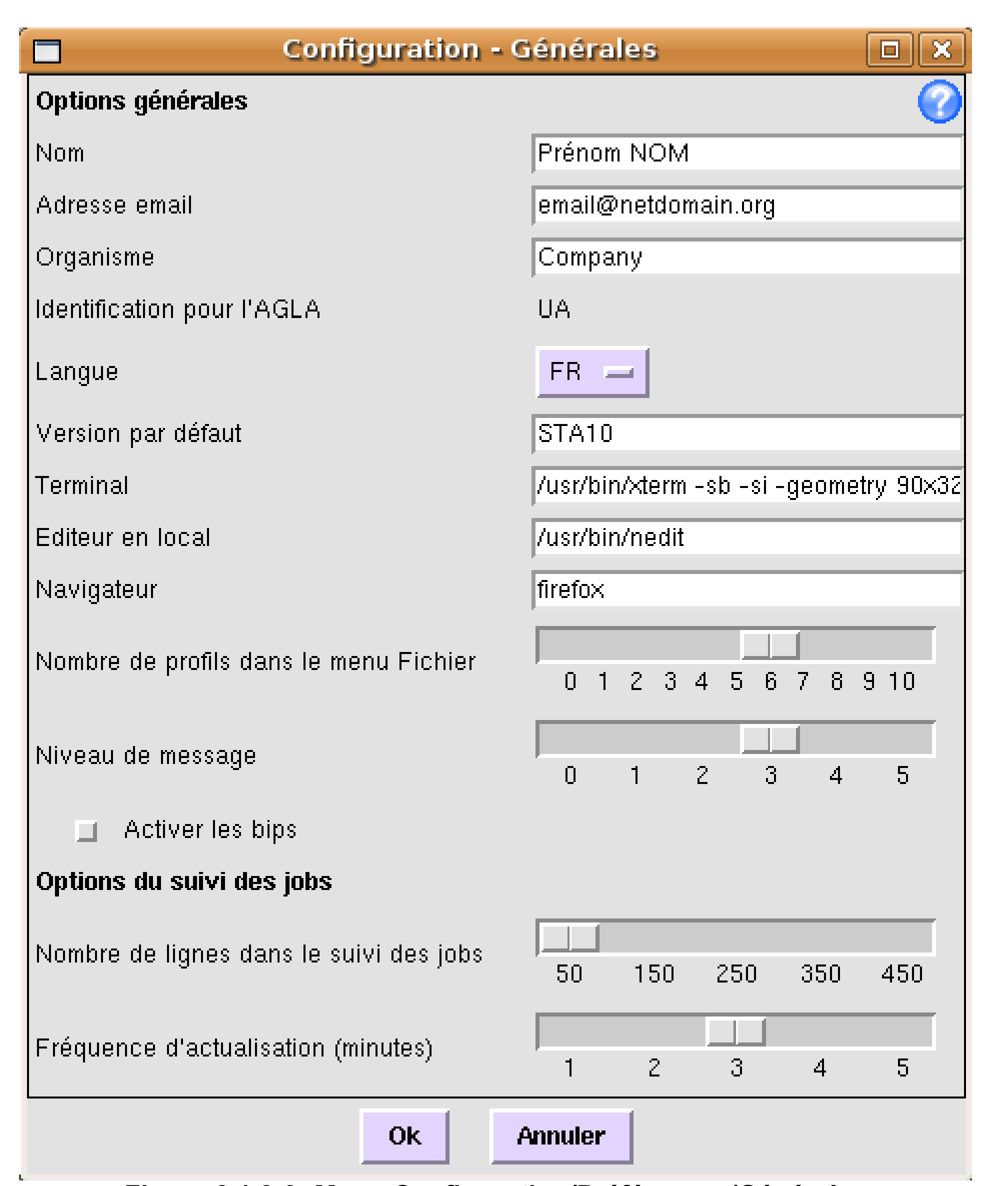


Figure 2.1.2-3: Menu Configuration/Préférences/Générales

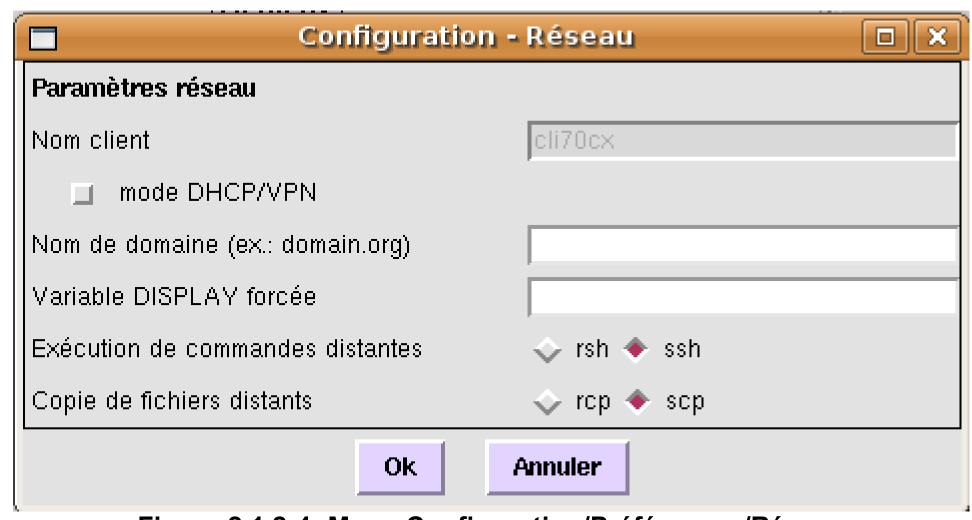


Figure 2.1.2-4: Menu Configuration/Préférences/Réseau

Remarques

Que l'on choisisse le protocole rsh/rcp ou ssh/scp, les connexions doivent fonctionner sans mot de passe : fichiers .rhosts correctement remplis ou clés privées/publiques SSH valides.
Selon ce que retourne la commande « uname » sur la machine, il arrive qu'astk pense être sur une machine distante et fasse alors un rsh ou ssh pour exécuter une commande locale... Dans ce cas, la connexion triviale de la machine vers elle-même doit également être configurée.

2.1.3 Menu Outils

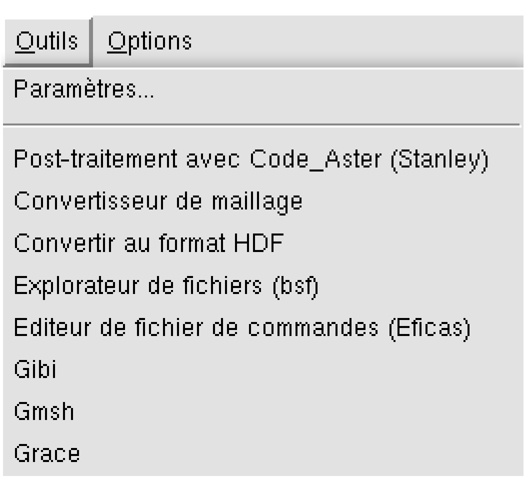


Figure 2.1.3-1: Menu Outils

- **Paramètres** : Permet de configurer le lancement des outils.
  Certains outils sont prévus en standard (sans forcément être disponibles sur la machine locale) :
  - `Post-traitement avec Code_Aster` : Outil de post-traitement, commande STANLEY,
  - `Convertisseur de maillage` : Permet de convertir un maillage d'un format vers un autre, les formats disponibles sont : `aster`, `med`, `gmsh`, `ideas` et `gibi`,
  - `Convertir au format HDF` : Permet de convertir une base classique Aster au format HDF,
  - `bsf` : Explorateur de fichiers multi-machines,
  - `Eficas` : Editeur de fichier de commandes *Aster*,
  - `Gibi` : Mailleur et outil de post-traitement (utilisable gratuitement avec *Aster*),
  - `Gmsh` : Mailleur et outil de post-traitement (libre),
  - `grace` : Traceur de courbes 2D.

Lorsque l'on choisit de lancer Stanley, astk cherche dans le profil les bases disponibles (par ordre de préférence une `base` en résultat, s'il n'y en a pas, une `bhdf` (base HDF) en résultat, sinon une `base` en donnée et finalement, une `bhdf` en donnée), produit un profil temporaire à partir du profil courant avec un fichier de commandes Aster qui commence par `POURSUITE()` et qui lance `STANLEY()`.

L'utilisateur peut ajouter ses propres outils, paramétrer le chemin d'accès aux outils (y compris modifier la commande d'accès aux outils standards), définir à quels types de fichiers associer l'outil et préciser si on peut utiliser l'outil sur un fichier distant.

Les codes suivants peuvent être utilisés dans la ligne de commande :
- @F : chemin absolu du fichier sélectionné,
- @R : répertoire contenant le fichier sélectionné,
- @f : nom du fichier (sans le répertoire),
- @D : adresse du DISPLAY (celui connu au moment du lancement de l'interface).

Les outils sont appelés soit à partir du menu Outils, soit à partir du menu contextuel sur un fichier d'une liste ou dans l'explorateur (bouton droit).
Pour sélectionner un fichier, il suffit de cliquer sur son nom dans une liste (dans l'onglet Etude, Tests ou Surcharge).
En double-cliquant sur un fichier, l’association entre le type du fichier et l’outil à utiliser s’appuie sur l’extension du nom de fichier pour la bsf, alors que le type sélectionné par la liste déroulante (cf. description des onglets) prévaut dans astk.

2.1.4 Menu Options

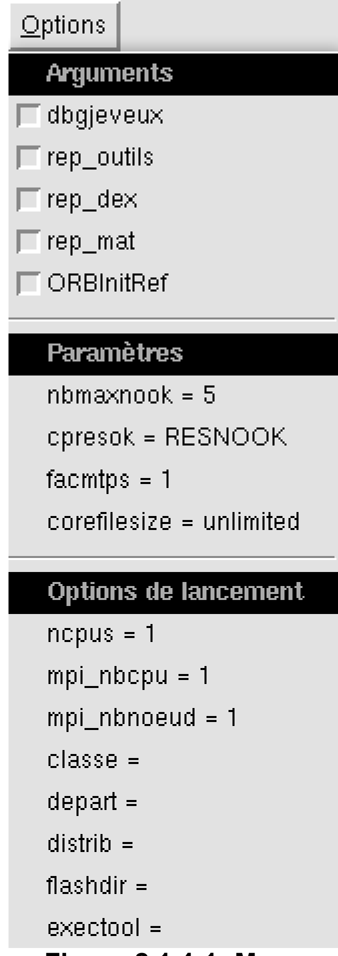


Figure 2.1.4-1: Menu Options

- Arguments :

  Permet de positionner des arguments optionnels qui seront passés sur la ligne de commande de l'exécution de *Code_Aster*.

  `dbgjeveux` : active un mode différent de la gestion des objets en mémoire pour détecter les écrasements et destruction d'objets,

  `rep_outils` : définit le répertoire où sont cherchés les outils appelés depuis *Aster* (onmetis ou gibi par exemple),

  `rep_dex` : définit le répertoire où sont cherchées les données externes (données de maillage),

  `rep_mat` : définit le répertoire où sont stockées les données du catalogue matériau.

  `ORBInitRef` : permet de contacter une session Salomé active (cette valeur est automatiquement positionnée quand astk est lancé depuis Salomé).

  Les arguments sélectionnés apparaissent dans la partie grisée de la zone des arguments passés à *Code_Aster*. La partie gauche de cette zone est libre.

- Paramètres
  Il s'agit de définir des paramètres optionnels qui seront écrits dans le fichier .export. Les trois premiers sont utilisés lors du lancement de tests.
  `nbmaxnook` : nombre maximal de cas-tests invalides (NOOK, ARRET_ANORMAL...) avant l'interruption de la liste des tests,
  `cpresok` : permet de choisir quels sont les tests dont on garde les fichiers résultat,
    RESOK : on garde les fichiers des tests OK et NOOK,
    RESNOOK : on ne garde que les fichiers des tests NOOK,
  `facmtps` : facteur multiplicatif du temps des tests (par rapport au temps de référence du para). Utile, par exemple, quand on soumet des tests avec un exécutable construit en mode debug, plus lent.
  `corefilesize:` taille limite du fichier core lors d’un plantage

- Options de lancement
  `ncpus :` définit le nombre de processeurs utilisés par le solveur MULT_FRONT
  `mpi_nbcpu :` définit le nombre de processeurs pour le parallélisme MPI (MUMPS, FETI)
  `mpi_nbnoeud :` définit le nombre de nœuds pour le parallélisme MPI (Où les `mpi_nbcpu` processeurs seront distribués)
  `classe :` permet de choisir la classe batch (ou le groupe de classe) dans laquelle le calcul sera soumis. Il faut bien évidemment vérifier que la classe existe et que les paramètres temps et mémoire sont compatibles avec cette classe.
  `depart :` permet de différer le départ d'un calcul. L'heure de départ est fournie au format `[[mois:]jour:]heure:minute`
  `distrib` : lance une étude paramétrique (valeur oui/non, voir §8).
  `flashdir` : pour les études paramétriques et le lancement de tests, permet de choisir un répertoire particulier pour les traces d'exécution des calculs ou None pour supprimer ces traces (voir §7).
  `exectool` : lancement en utilisant un outil particulier (voir §5.6.1)

2.1.5 Menu Aide

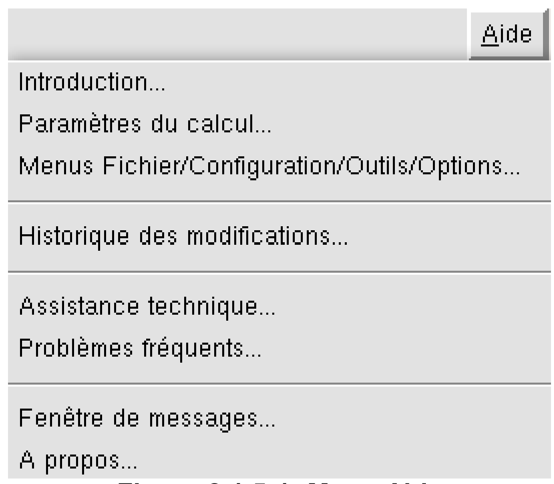


Figure 2.1.5-1: Menu Aide

- Introduction/Paramètres du calcul/Menus : accès au texte d'aide
- Paramètres de calculs : permet de définir la taille mémoire, la version de Code_Aster qui sera utilisées, le mode de calcul (Batch ou interractif), …
- Menus Fichier/Configuration/outils/options…
- Historique des modifications : Evolution d'astk au fil des versions, nouvelles possibilités, corrections effectuées, anomalies connues...
- Assistance Technique…
- Problèmes fréquents : Quelques questions fréquemment posées avec leurs solutions.

- Fenêtre des messages : Les messages d'informations <INFO>, d'erreurs <ERREUR> sont écrits dans cette fenêtre. Avec un niveau de message supérieur ou égal à 1 (voir [§ 2.1.2]), on obtient plus ou moins d'informations.
- A propos… : l'incontournable fenêtre d'information.

2.2 Les onglets

On trouve cinq boutons dans cette zone. On parle d’onglet quand le contenu que l’on trouve sous le bouton change selon celui qui est pressé. Graphiquement, ce ne sont pas de « vrais » onglets car le widget n’existe pas dans la librairie Tk standard (et on n’a pas voulu ajouter un pré-requis supplémentaire !).

ETUDE, TESTS et SURCHARGE sont des onglets dans lesquels on renseigne la liste des fichiers nécessaires ; AGLA et REX sont des boutons qui peuvent utiliser le contenu des autres onglets.

Enfin, la case à cocher située à côté des quatre premiers boutons signale que l’on utilise (ou non) le contenu de l’onglet associé.
Exemples : Pour lancer une étude, il faut cocher l’onglet ETUDE. Pour faire appel aux fonctions de l’AGLA (sur la machine de référence), il faut cocher l’onglet AGLA ; on notera que dans ce cas, TESTS et SURCHARGE sont automatiquement cochés car leur contenu est nécessairement pris en compte.

2.2.1 Onglets ETUDE, TESTS, SURCHARGE

Pour accéder plus facilement aux fichiers, parcourir les arborescences plus rapidement (les fichiers étant souvent regroupés dans des répertoires proches), ou encore simplifier l’affichage des noms de fichiers, on peut définir un **chemin de base**.

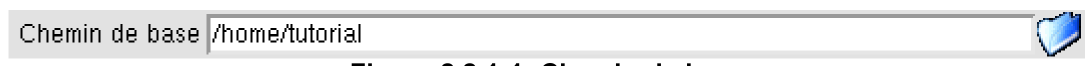


Figure 2.2.1-1: Chemin de base

On le définit en cliquant sur le bouton Parcourir et on choisit le répertoire qui sera le répertoire par défaut.

Chaque onglet contient une liste de fichiers (deux listes pour SURCHARGE).



Figure 2.2.1-2: Liste de fichiers et répertoires

On définit un fichier ou répertoire par ligne. On trouve de la gauche vers la droite :

- Type : type du fichier ou du répertoire ;
- Serveur : nom du serveur sur lequel se trouve le fichier/répertoire ;
- Nom : chemin d’accès au fichier : en absolu s’il commence par ‘/’, en relatif par rapport au chemin de base dans le cas contraire ;
- UL : numéro d’unité logique associé à ce fichier ;
- D, R, C : cases à cocher pour indiquer si le fichier/répertoire est respectivement en donnée, résultat, compressé (avec gzip).

Lorsque l’on change le type d’un fichier/répertoire, des valeurs par défaut sont positionnées pour les indicateurs D, R, C et le numéro d’unité logique.

Remarque sur les types :

Code_Aster manipule les fichiers via les procédures fortran qui utilisent des numéros d'unité logique (fichier `fort.19` par exemple) ; il affecte donc des numéros d'unité logique par défaut pour simplifier la vie de l'utilisateur. Le « type » permet ainsi d'accéder facilement aux numéros utilisés par défaut ; il permet aussi à astk de vérifier que l'utilisateur fournit des données cohérentes (par exemple que l'on fournit bien un répertoire pour tel type.

Pour manipuler la liste, on dispose de six boutons :

- : ajoute une nouvelle entrée vierge à la fin de la liste ;
- : ajoute un fichier/répertoire à la fin de la liste en parcourant l'arborescence de fichiers ;
- : supprime la ligne actuellement sélectionnée dans la liste. Un clic-droit permet de supprimer toutes les lignes ;
- : ouvre le fichier/répertoire actuellement sélectionné dans la liste avec l'éditeur du serveur où se trouve le fichier (cf. [§ 4.1]). S'il s'agit d'un répertoire, tous les fichiers du répertoire sont ouverts avec l'éditeur (attention aux répertoires contenant beaucoup de fichiers ou des fichiers binaires non éditables !) ;
- : déplace une ligne vers le haut ;
- : déplace une ligne vers le bas.

Arguments (onglet ETUDE et TESTS seulement) :
Cette zone de texte permet de transmettre des arguments à l'exécutable *Aster*. Voir aussi [§2.1.4].

Menu contextuel :
En cliquant avec le bouton droit sur une entrée de la liste, on accède à un menu contextuel :

- Ouvrir : lance l'outil associé à ce type de fichier (déterminé par le champ « Type », et non l'extension du fichier), si aucun outil n'est associé à ce type, on édite le fichier ;
- Editer : édite le fichier (ou tous les fichiers du répertoire) de la même manière que le bouton  ;
- Imprimer : imprime le fichier sélectionné ;
- Ouvrir avec… : on peut choisir d'ouvrir le fichier sélectionné avec un des outils disponibles (le résultat peut être étonnant si l'outil ne connaît pas ce type de fichier !) ;
- Valeurs par défaut : l'interface détermine un nom de fichier par défaut en fonction du « Type » choisi à partir du nom du profil (fichier `.astk`), aucune valeur n'est proposée si le profil n'a pas encore été enregistré (s'utilise en général sur une ligne vierge que l'on vient d'insérer, les indicateurs D/R/C ne sont pas affectés par cette opération) ;
- Terminal : permet d'ouvrir une fenêtre terminale `xterm`
- Propriétés : affiche les permissions, la taille, la date et l'heure du fichier (commande `ls -la`).

Onglet SURCHARGE

Les données (fichiers sources) sont fournies dans la liste supérieure, alors que les résultats (exécutable, catalogues compilés) sont fournis dans la liste inférieure.

2.2.1.1 Liste des types pour ETUDE

comm :	fichiers de commande *Aster* (y compris les fichiers de poursuite)
mail :	fichier maillage au format *Aster*
erre :	fichier d'erreur (fort.9 d'Aster)
mess :	fichier des messages de l'exécution
resu :	fichier de résultat (impression des tests, impression au format *Aster*)
base :	répertoire contenant la base du calcul
bhdf	répertoire contenant la base du calcul au format HDF
cast :	fichier résultat au format CASTEM
mast :	gardé pour raison de compatibilité
mgib :	maillage au format Gibi
mmed :	maillage au format MED
msh :	maillage au format Gmsh
msup :	maillage au format IDEAS
datg :	fichier de commande Gibi
pos :	fichier résultat au format Gmsh
ensi :	répertoire résultat au format Ensight
dat :	fichier résultat contenant des courbes au format XMGRACE
ps :	fichier au format postscript
agraf :	fichier résultat contenant les données pour Agraf (les anciennes versions d'Aster écrivait les directives et les données dans un même fichier qu'il fallait découper avec la commande `post_agraf` sur la machine de référence)
digr :	fichier résultat contenant les directives pour Agraf
rmed :	fichier résultat au format MED
unv :	fichier résultat au format « UNiVersel » (IDEAS)
distr :	fichier des valeurs utilisées pour une étude paramétrique
hostfile :	fichier décrivant les ressources machines à utiliser (étude paramétrique)
nom :	pour récupérer à partir de son nom, un fichier présent dans le répertoire temporaire /tmp
para :	fichier de paramètres (retranscription des paramètres du calcul pour les tests)
repe :	répertoire en données et/ou résultats (permet de transmettre/récupérer le contenu complet d'un répertoire ; comme on n'affecte pas de numéros d'unité logique *Aster* doit accéder aux fichiers par leurs noms). répertoire des résultats lors d'une étude paramétrique
libr :	fichier ou répertoire au choix de l'utilisateur
btc :	script de lancement généré par le service (on peut ainsi le récupérer, le modifier…).

Lors d'un astout sur la machine de référence, `resu_test` doit être sur celle-ci.
Lors d'une étude paramétrique, le répertoire des résultats (type `repe`) doit être sur la machine d'exécution.

2.2.1.2 Liste des types pour TESTS

list :	fichier contenant la liste des tests à exécuter (un nom de test par ligne sans `.comm`)
rep_test :	répertoire contenant les fichiers de données des tests (commandes, maillage…)
resu_test :	répertoire où sont recopiés les fichiers résultats
btc :	idem ETUDE
hostfile :	fichier décrivant les ressources machines à utiliser

2.2.1.3 Liste des types pour SURCHARGE, partie DONNEES

f :	sources fortran
f90	sources fortran 90
c :	sources C
py :	sources Python
capy :	sources des catalogues de commandes
cata :	sources des catalogues d'éléments, d'options, de grandeurs…
hist :	fichier histor (historique des modifications)
conf :	fichier de configuration (choix des options de compilation, des librairies…)

unig : fichier `unigest` contenant les modules à supprimer
datg : données géométriques
cmat : catalogues de données matériaux

En général, on fournit un répertoire pour les cinq premiers types (sources) ; tous les fichiers dont l'extension correspond au type indiqué sont pris en compte. Néanmoins, il est déconseillé de mélanger les types de fichiers dans un même répertoire.
hist, unig, datg et cmat sont exclusivement liés à la gestion de la version de référence.

2.2.1.4 Liste des types pour SURCHARGE, partie RESULTATS

exec : fichier de l'exécutable *Aster*
cmde : répertoire du catalogue de commande compilé
ele : fichier du catalogue d'éléments
btc : idem ETUDE

2.2.2 Bouton AGLA

Ce bouton permet d'accéder aux fonctions de l'atelier de génie logiciel qui coordonne les actions des développeurs de la version de cohérence. Pour cela, la case à droite du bouton doit être cochée, ce qui a pour effet de prendre en compte automatiquement le contenu des onglets TESTS et SURCHARGE, il suffit ensuite de cliquer sur le bouton « Lancer ».
On se reportera au manuel de l'AGLA ([D1.02.01]) pour plus de détails sur les différentes actions.

- **ASNO** : permet de « noter » des modules (signaler que l'on prévoit de restituer une modification), s'applique aux fichiers sources et aux fichiers de test.
- **ASDENO** : permet de « dénoter » des modules. Cette action n'utilise aucune des données de TESTS ou SURCHARGE. Lorsque l'on clique sur « Lancer », une fenêtre demande de choisir le type de module à dénoter (fortran/C, python, catalogues ou test), et d'indiquer le nom des modules séparés par un espace, une tabulation ou un retour à la ligne (**sans l'extension** : `op0191` pour dénoter le fichier `op0191.f`). Pour les fichiers Python, il est nécessaire de préciser le nom du répertoire (package dans la dénomination Python) dans lequel ils se trouvent (car contrairement aux sources fortran et C, le même nom peut être utilisé dans des répertoires différents), exemple : `macr_recal_ops@Macro` (désigne le module `macr_recal_ops` du package `Macro`).

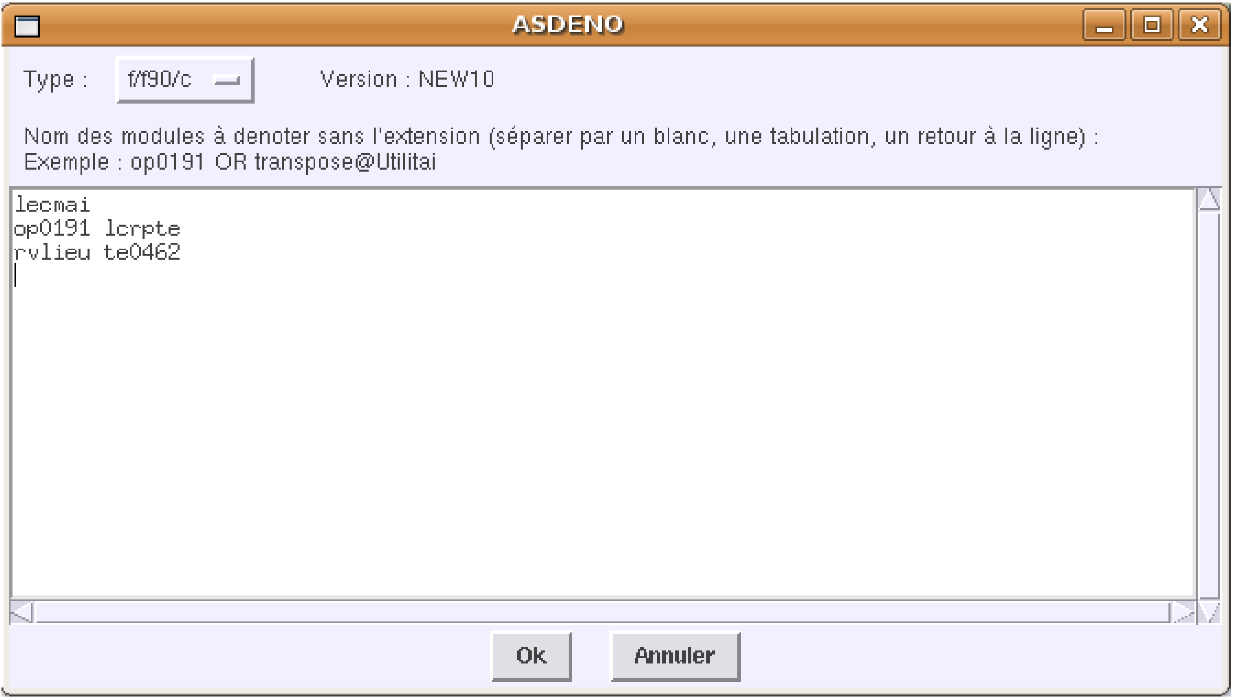


Figure 2.2.2-1: fenêtre ASDENO

- **ASQUIT** : fournit dans le fichier output la liste de tous les modules notés.
- **ASVERIF** : vérifie qu'un ensemble de sources peut être intégré dans la version de référence (respect des règles de programmation, fichier histor présent…).
- **PRE_EDA** : il doit être effectué avant la réunion de l'équipe de développement, le développeur signale ainsi son intention de restituer des sources, PRE_EDA soulève les points qui devront être réglés en réunion de développement (PRE_EDA effectue un ASVERIF et un passage des tests fournis dans le profil).
- **ASREST** : il s'agit de la phase ultime de la restitution qui consiste en un ASVERIF, un passage des tests de la liste de non régression, le code retour doit être inférieure ou égal à 2 pour que la restitution soit prise en compte.

Remarque

Les fonctionnalités AGLA ne sont disponibles que sur le serveur de référence.

2.2.3 Bouton REX

Ce bouton permet :

- D'**Emettre une fiche sans joindre le profil** : Cette fonction permet à l'utilisateur d'émettre une fiche de retour d'expérience pour signaler une anomalie dans *Code_Aster* (AL : anomalie logiciel), demander une évolution de *Code_Aster* (EL : évolution logiciel), dans un des outils associés (Metis, Homard, Eficas, astk, bsf…) (AO : anomalie outil/EO : évolution outil), une modification de la documentation (ED : évolution documentation), ou une demande d'expertise en modélisation (AOM : aide à l'optimisation de la modélisation). Les informations relatives à l'émetteur de la fiche sont accessibles par le menu *Configuration/Interface*. Les fichiers contenus dans le profil courant ne sont pas joints à la fiche, ce qui peut rendre très difficile le traitement de la fiche. Il est conseillé d'utiliser "émettre une fiche et y associer les fichiers". La version de *Code_Aster* indiquée est celle sélectionnée dans l'interface (*Paramètres du calcul*).
- D'**Emettre une fiche et y associer les fichiers** : Cette fonction permet d'émettre le même type de fiche que la précédente, cette fois-ci les données incluses dans le profil sont jointes à la fiche. Pour une anomalie, les fichiers pour reproduire l'erreur devraient toujours être fournis. Ce qui n'empêche pas l'utilisateur d'essayer d'isoler autant que

possible le problème rencontré, notamment d'essayer de joindre une étude sur un modèle qui nécessite peu de mémoire et de temps de calcul !
- De **Consulter les fiches** : permet d'accéder à l'outil de gestion de retour d'expérience (sur la machine de référence).

Remarque

Les fonctionnalités REX ne sont disponibles que sur le serveur de référence.

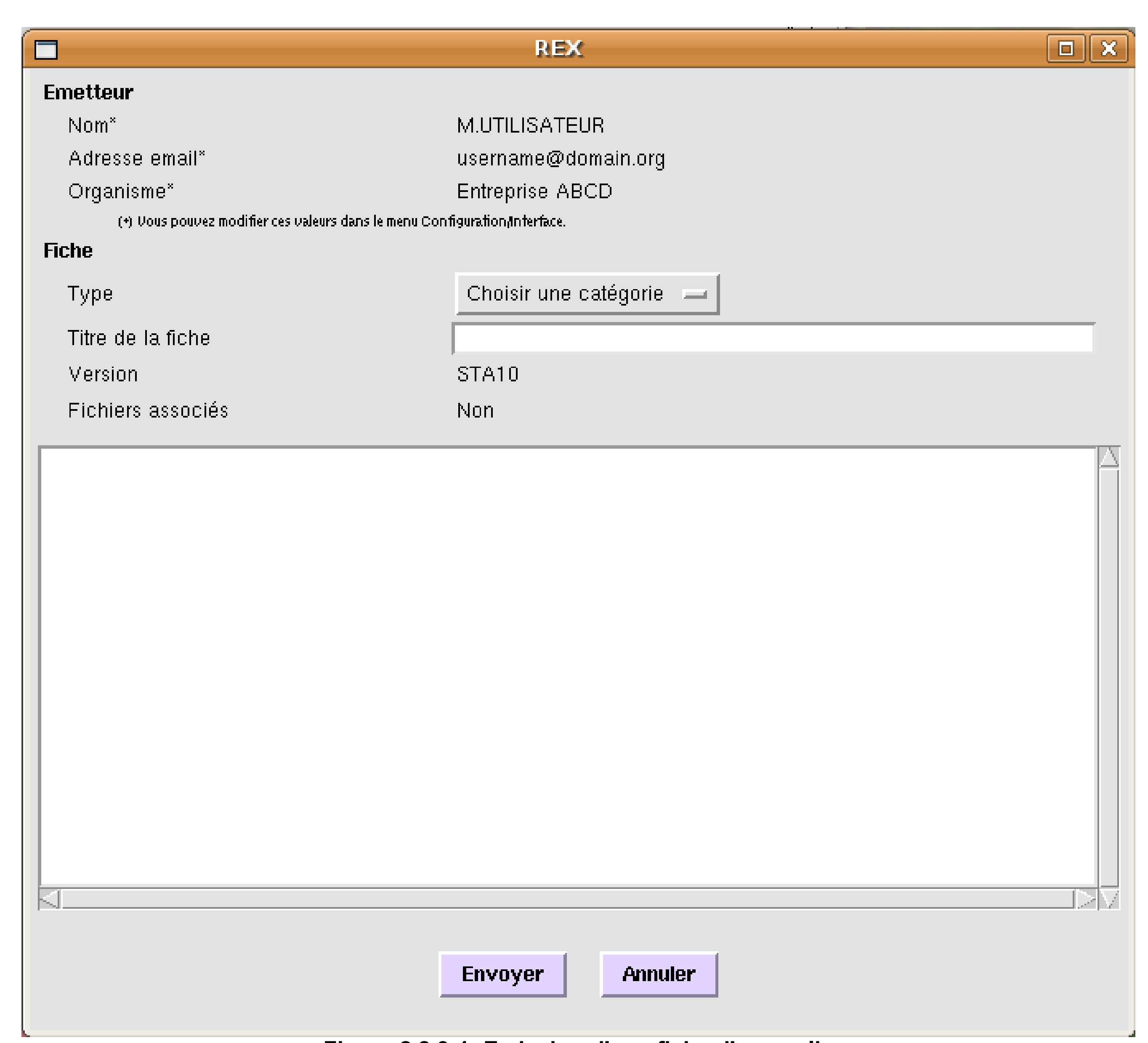


Figure 2.2.3-1: Emission d'une fiche d'anomalie

2.3 Paramètres de soumission

Les paramètres du calcul sont fournis dans la partie droite de la fenêtre principale.

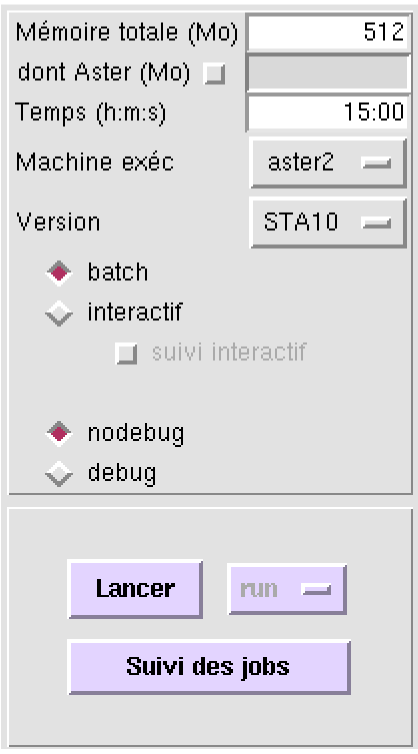


Figure 2.3-1: Zones des paramètres du calcul

On définit la quantité de :

- `Mémoire totale` utilisée pour le job (en mégaoctets).
- `Dont Aster` : limite la mémoire utilisée par Aster (et donc laisse de la mémoire pour d'autres produits). Par défaut la mémoire aster est égale à 90% de la mémoire totale.
- le temps maximum du calcul (en secondes, minutes:secondes ou heure:minutes:secondes).

On choisit sur quelle machine le calcul est exécuté, la version de *Code_Aster* utilisée, si le calcul est soumis en batch ou en interactif.
debug/nodebug : pour une étude sans surcharge, on précise quel exécutable on souhaite utiliser (sous réserve que les deux soient disponibles) ; lors d'une surcharge, on choisit de compiler avec ou sans les informations de debug.

Le bouton « Lancer » exécute les actions en fonction des onglets cochés.
Le bouton « Suivi des jobs » ouvre la fenêtre décrite après.

Mode de fonctionnement (pour une ETUDE) :
Lors du lancement d'une étude (avec ou sans surcharge), un bouton d'option est disponible à côté du bouton « Lancer ». Trois modes de lancement sont disponibles :

- « run » : exécute l'étude (fonctionnement classique),
- « dbg » : lance l'étude en utilisant le debugger,
- « pre » : prépare le répertoire de travail sans exécuter l'étude.

Quand on sélectionne « dbg » ou « pre », le mode « debug » est choisi par défaut.

2.4 Barre d'état

La zone de texte située tout en bas de la fenêtre principale fournit de l'aide lorsque l'on navigue dans les menus, ou bien quand le pointeur passe au dessus des boutons de l'interface.
Lors du lancement d'un calcul, les opérations en cours sont affichées ici.

Groupe de classe batch (laisser vide pour utiliser les classes par défaut)
Valeurs possibles : voir Configuration/Serveurs.

Figure 2.4-1: Barre d'état

2.5 Suivi des jobs

Cette fenêtre fournit des informations sur l'état des calculs lancés, qui apparaissent sous forme d'une liste. En cliquant avec le bouton droit sur un calcul, on a les choix suivants :

- **Editer** se décompose en « Fichier output », « Fichier error » et « Fichiers export » qui donne accès aux messages envoyés par le job sur la sortie standard (`stdout`) et la sortie d'erreur (`stderr`). On peut aussi accéder au fichier .export utilisé lors de l'exécution. Un double-clic sur un job de la liste affiche également le fichier output.
- **Supprimer** efface les jobs sélectionnés de la liste, les fichiers liés à ce job dans le flasheur et interrompt le calcul si celui-ci n'est pas terminé (en envoyant le signal SIGKILL).
- **Interrompre USR1** envoie le signal SIGUSR1 au calcul. Code_Aster intercepte ce signal pour interrompre le calcul, puis ferme proprement la base qui sera rapatriée (si une base est en résultat dans le profil).
- **Actualiser** interroge les serveurs sur lesquels des calculs sont en cours. La case à cocher permet de réactualiser automatiquement à une fréquence définie dans *Configuration/Interface*.
- **Actualiser tous** rafraichit l'état de tous les calculs non terminés.
- **Rechercher** permet de consulter les dernières lignes du fichier message d'un job *en cours d'exécution* (il ne fait rien sur un job terminé). On peut utiliser la zone de texte « Filtre » pour n'afficher que les lignes contenant la chaîne de caractères indiquée (sous forme d'expression régulière).

Chaque ligne correspond à un job, on trouve 13 colonnes :

- Le numéro du job (en batch), numéro du processus en interactif
- Le nom du job (nom du profil pour une étude, une surcharge, ou nom de la fonction AGLA)
- Date de soumission
- Heure de soumission
- Etat du job (PEND, RUN, SUSPENDED, ENDED)
- Diagnostic du job (OK, NOOK, <A>_ALARM, <F>_ERROR, <F>_ABNORMAL…)
- Nom de la queue en batch ou « interactif »
- Temps CPU de l'exécution *Aster*
- Login sur le serveur de calcul utilisé
- Adresse du serveur de calcul utilisé
- Machine de calcul (nom du nœud pour un cluster)
- Version d'astk
- Indicateur batch/interactif

3 Boite de sélection de fichiers : bsf

bsf est un outil livré avec astk qui peut être lancé seul. Il s'agit d'un explorateur de fichiers qui permet de naviguer sur la machine locale, comme un explorateur de fichiers classique, et aussi sur les différents serveurs distants configurés.

bsf utilise la configuration des serveurs de astk, notamment les champs nécessaires à la connexion (adresse IP, login) et les commandes pour ouvrir un terminal ou un éditeur.
La lecture de la configuration n'est faite qu'au démarrage de bsf, si on modifie la configuration dans astk, il faut donc fermer la bsf puis l'ouvrir de nouveau.

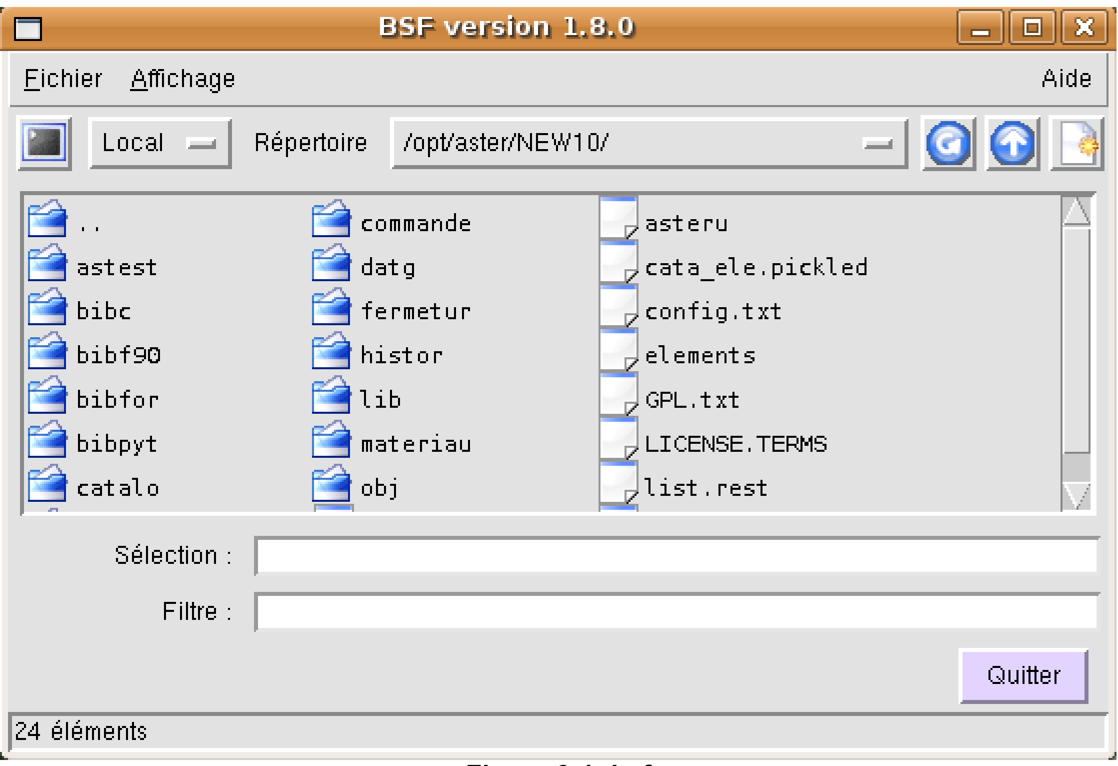


Figure 3-1: bsf

Remarque :

> *bsf ne traite pas les noms de fichiers et répertoires contenant les espaces (il en résulte un décalage dans les noms et les types des objets suivants).*

3.1 Navigation

On trouve deux listes déroulantes dans la fenêtre de la bsf.
La première permet de passer d'un serveur à un autre, la deuxième mémorise la liste des douze derniers répertoires où l'utilisateur a effectué une action (édition, copie…). La première entrée de cette liste est '----- Direct -----', qui permet d'aller directement dans un répertoire dont on saisit le nom.
D'une manière générale, lorsque le répertoire demandé n'existe pas, on retourne dans le HOME défini pour le serveur courant.
On peut configurer cette liste et fixer certains répertoires (pour qu'ils restent présents dans la liste) en cliquant avec le bouton droit sur celle-ci (cf. Figure 3.1-a).

- : rafraîchit le contenu du répertoire courant ;
- : remonte au répertoire parent ;
- : propose de créer un nouveau répertoire dans le répertoire courant (et se place dans ce nouveau répertoire) ;
- : ouvre un terminal sur le serveur actuel.

La barre d'état donne des indications sur la signification de ces boutons lorsque le pointeur de la souris passe au dessus.

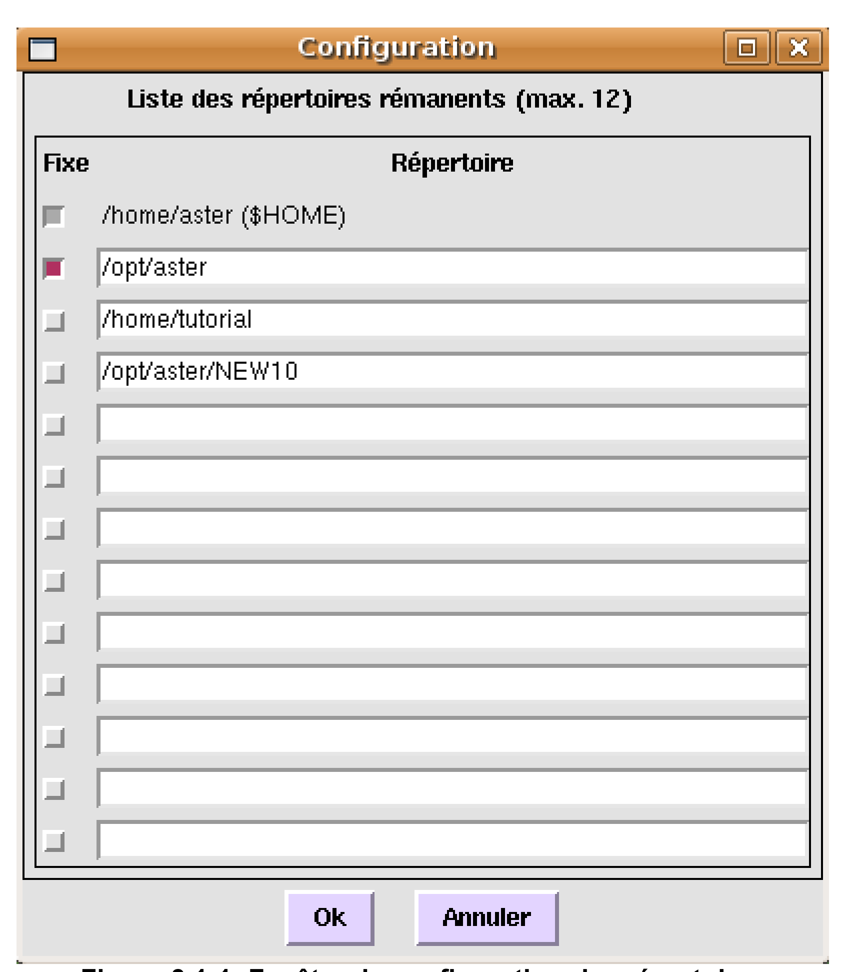


Figure 3.1-1: Fenêtre de configuration des répertoires rémanents

3.2 Menu Fichier et menu contextuel

On retrouve certaines entrées semblables au menu contextuel des fichiers dans les onglets (voir [§2.2.1]) : Ouvrir, Editer, Imprimer, Ouvrir avec…, Propriétés.

- **Copier** : permet de copier un ou plusieurs fichiers/répertoires (raccourci par CTRL+C).
- **Couper** : idem Copier, sauf que les fichiers originaux seront supprimés (raccourci par CTRL-X).
- **Coller** : réalise effectivement la copie ou le déplacement des fichiers (raccourci par CTRL+V).
- **Supprimer** : efface les fichiers/répertoires sélectionnés.
- **Renommer** : donne un nouveau nom à un fichier
- **Nouveau fichier/Nouveau répertoire** : permettent respectivement de créer un fichier ou un répertoire dans le répertoire courant.
- **Exécuter une commande** : donne la possibilité d'exécuter une commande dans le répertoire courant, la sélection courante est proposée sur la ligne de commande, il faut choisir un des shells disponibles sur le serveur.
- **Terminal** : ouvre un terminal sur le serveur actuel.

3.3 Menu Affichage

On peut choisir d'afficher ou non les fichiers dits cachés (commençant par '`.`') en cochant la case prévue.
bsf affiche les répertoires puis les fichiers, on peut les trier par ordre alphabétique ou en fonction de la date dans l'ordre normal ou inverse en modifiant les options du menu.

Les préférences d'affichage sont conservées si l'on sélectionne **Enregistrer les préférences**.

4 Configuration

La première fois que l'utilisateur lance l'interface, il dispose d'une configuration par défaut qui a été définie lors de l'installation. La configuration est ensuite stockée dans le répertoire `$HOME/.astkrc`. Si l'utilisateur veut revenir à la configuration d'origine, il doit effacer ce répertoire et relancer astk.

Remarque n°1

> *A chaque fois que l'on définit une commande à exécuter (terminal, éditeur…), il est conseillé d'indiquer le chemin absolu (depuis la racine) pour éviter que la commande ne soit pas trouvée si la variable `$PATH` est incorrecte.*

Remarque n°2

> *En général, astk complète les commandes qui ont besoin d'afficher des fenêtres avec l'argument ' `-display $DISPLAY` '. Si la commande utilisée n'accepte pas cet argument, il suffit de préciser la bonne syntaxe avec le code @D qui sera remplacé par la variable courante de `$DISPLAY` (par exemple : gedit `--display=@D` , cf. Menu Outils).*
> *On peut vérifier la valeur courante de `$DISPLAY` , menu Aide/Fenêtre des messages : astk indique dans les premières lignes par exemple :*
> ```
> <INFO> Affichage des applications sur mach00.domain.org:0.0
> ```
> *C'est cette valeur qui sera fournie aux machines distantes sensées afficher une fenêtre sur la machine cliente.*

La partie as_run contient des outils simplifiant certaines tâches des développeurs comme la consultation du code source ou la mise à jour d'une version locale de développement.
Le fichier `$HOME/.astkrc/config` contient donc deux informations nécessaires pour contacter le serveur de référence : le nom complet de ce serveur et le login de l'utilisateur. Ce dernier doit être défini dans ce fichier pour éviter l'alarme suivante :

```
<A>_ALARM          remote connection may fail :
devel_server_user not defined in /home/xxxxxx/.astkrc/config
```

4.1 Serveurs

On accède à la fenêtre de configuration par le menu *Configuration/Serveurs* (voir Figure 2.1.2-2).

Le premier bouton permet de passer d’un serveur à un autre et d’ajouter un « Nouveau serveur ».
Les champs sont :

- Nom complet ou adresse IP : il s’agit du nom du serveur sur le réseau ; on peut indiquer son nom complet avec le nom de domaine (par exemple : linux.labo.univ.fr) ou son adresse IP (par exemple : 156.98.254.36).
- La case indique que astk ne doit pas utiliser rsh ou ssh pour contacter cette machine.
- Etat du serveur : on peut mettre sur « off » un serveur temporairement inaccessible.
- Login : identifiant avec lequel on se connecte au serveur.
- Répertoire HOME : répertoire par défaut lorsque l’on navigue sur cette machine.
- Répertoire des services : répertoire où sont installés les services sur ce serveur (indiquer le chemin d'installation, par exemple : `/opt/aster`), laisser vide pour un serveur de fichiers.
- Mode de téléchargement de la configuration : aucun (pour un serveur de fichiers), manuel (il faut cliquer sur le bouton « Télécharger maintenant » pour récupérer la configuration Aster du serveur), automatique (astk interroge le serveur au démarrage tous les 30 jours).
- Dernier téléchargement : date de la dernière mise à jour des informations de configuration.
- Terminal : commande pour ouvrir un terminal sur le serveur. Ceci permet d’ouvrir une fenêtre de commandes sur le serveur quand on utilise la bsf, et de suivre interactivement un calcul (le terminal doit accepter l’option `-e` pour exécuter une commande, `xterm` convient).
- Editeur : éditeur texte (par exemple, `nedit`). La procédure d’installation choisit un éditeur parmi (et dans cet ordre) : `nedit`, `gedit`, `xemacs`, `emacs`, `xedit`, `vi`.

Les valeurs suivantes sont retournées par `as_run --info` (rien pour un serveur de fichiers) et dépendent donc de la configuration de la partie as_run :

- Type de plate-forme.
- Versions disponibles.
- Machines accessibles : liste des nœuds de calcul accessibles depuis ce serveur.
- Batch/interactif : précise si le serveur accepte le lancement en batch, en interactif et fournit les limites en mémoire, temps CPU, nombre de processeurs fixés sur le serveur.

Les logiciels de gestion de batch supportés sont LSF, Sun Grid Engine et PBS.

4.2 Configurations

4.2.1 Préférences Générales

On accède à la fenêtre de configuration des préférences générales par le menu *Configuration/ Préférences/générales* (voir Figure 2.1.2-3).

Cette fenêtre permet de renseigner les informations personnelles de l’utilisateur, de choisir la langue utilisée par l’interface.
Pour ceux qui ont accès à la machine de référence, l’instance AGLA est affichée (EDA pour développeur, UTL pour utilisateur…). Pour les développeurs, l’organisme et le nom du correspondant sont automatiquement remplis. Les utilisateurs doivent le faire eux-mêmes.

Ensuite, on trouve la version qui sera sélectionnée par défaut, les commandes pour accéder à un terminal et un éditeur (comme pour les serveurs).
Nombre de profils dans le menu Fichier permet de conserver le nom des N derniers profils ouverts de manière à les rappeler rapidement.

Niveau de message indique le niveau de détails des messages écrits dans la fenêtre des messages du menu *Aide*. Niveau=0 : seuls les messages `<INFO>` et `<ERREUR>` sont écrits ; les niveaux supérieurs permettent de déboguer le comportement de l’interface. Le niveau 1 est conseillé, il permet de voir les messages d’erreur pouvant apparaître lors des problèmes de communication avec les serveurs distants.

Pour le suivi des jobs, on peut choisir le nombre de lignes affichées lorsque l’on visualise le fichier output en cours de job (bouton **Rechercher**), et la fréquence d’actualisation de la liste.

4.2.2 Préférences Réseau

On accède à la fenêtre de configuration des préférences concernant le réseau par le menu *Configuration/ Préférences/Réseau* (voir Figure 2.1.2-4).

Cette fenêtre permet de renseigner les paramètres réseau. On doit ensuite préciser le nom de domaine réseau de la machine. Par exemple, `domain.org` si le nom complet de la machine est `mach00.domain.org`. Si le nom de domaine est laissé vide, un message d'alarme est affiché au démarrage car les machines distantes risquent de ne pas pouvoir contacter la machine locale `mach00` avec son nom court. Le nom de la machine tel qu'il sera utilisé par les machines distantes est indiqué dans le champ Nom Client. Si le mode DHCP/VPN est actif, c'est dans ce champ que l'on peut indiquer l'adresse IP de la machine locale. En cliquant sur OK, l'interface propose les adresses IP des interfaces réseaux détectées sur la machine.

4.3 Outils

Voir Figure 2.1.3-1.

On sélectionne l'outil à configurer avec la liste déroulante, ou bien on ajoute un nouvel outil.
Des outils standards sont pré-définis (les minuscules/majuscules sont prises en compte dans les noms d'outils). *Les outils sont nécessairement exécutés sur la machine « Local »* (où est lancée l'interface).

En général, les outils sont lancés soit sur un fichier d'un onglet (ETUDE, TESTS ou SURCHARGE), soit sur un fichier lorsque l'on parcourt le système de fichiers avec la bsf.

On définit simplement la ligne de commande nécessaire au lancement d'un outil (chemin absolu conseillé), on peut placer les codes @F, @R, @f, @D dans la ligne de commande (voir [§ 2.1.3]) pour passer correctement un fichier à l'outil. On peut mettre ces codes entre parenthèses pour pouvoir lancer l'outil seul, sans fichier en argument.

Des types de fichiers peuvent être associés à l'outil. Le type pris en compte pour lancer l'outil est l'extension lorsque l'on parcourt les fichiers avec la bsf, le type de la liste déroulante quand il s'agit d'un onglet.

On peut choisir si l'outil est utilisable sur un fichier distant. Dans ce cas, astk se charge de ramener le fichier en question sur la machine locale dans un répertoire temporaire, de lancer l'outil, puis de redéposer le fichier sur le serveur distant (même s'il n'a pas été modifié par l'outil).

4.4 Impression

Les commandes d'impression sont définies par le menu *Fichier/Impression*.
Sur le même principe que les outils, on définit n'importe quelle commande d'impression (lp, lpr, a2ps…) qui contient le code correspondant au nom du fichier (@F, voir [§ 2.1.3]). @P est remplacé par le nom de la file d'impression.

Voici une liste de commande « classique » utilisant GNU a2ps :

Format 2 colonnes/page :
```
a2ps -P @P -2 -l82 --footer --print-anyway=yes --sides=tumble @F
```

Format 136 caractères/ligne :
```
a2ps -P @P -r -l136 --columns=1 --footer --print-anyway=yes --sides=tumble @F
```

Format "transparent" :
```
a2ps -P @P -r --columns=1 --print-anyway=yes --sides=simplex @F
```

5 Comment faire une étude ?

Dans ce paragraphe, on décrit étape par étape comment utiliser astk pour réaliser une étude.

L'étude consiste à calculer la réponse en traction d'une plaque trouée en adaptant le maillage. On dispose des éléments suivants :

- Le fichier de commande *Aster* : `demo001a.comm`
- La description de la géométrie réalisée avec Gmsh : `demo001.datg`
- Le maillage grossier de la plaque construit par Gmsh : `demo001a.msh`
- Le maillage fin du contour de la plaque : `demo001a.18`

On produit les résultats suivants :

- Un fichier de maillage *Aster* : `demo001a.mail`
- Un fichier résultat au format Gmsh (champs de déplacements, d'erreur…) : `demo001a.pos`
- Une courbe au format png (produite par xmgrace) : `demo001a.png`
- Les fichiers classiques de message et résultat Aster : `demo001a.mess` et `demo001a.resu`

Dans l'exemple, on place tous les fichiers dans le répertoire `/home/tutorial`.
Les fichiers de cet exemple sont disponibles dans le répertoire `astest` de votre version de *Code_Aster*, par exemple dans `/opt/aster/STA10.1/astest`.

Remarque :

> *Dans le cas d'une étude avec plusieurs fichiers de commandes, tous les fichiers doivent être de type « comm », associés à l'unité logique 1 et c'est l'ordre d'apparition dans le profil qui détermine l'ordre d'exécution.*

5.1 Création du profil

On lance l'interface qui s'ouvre sur un profil vierge, ou bien si astk est déjà lancé, on choisit *Fichier/Nouveau* dans le menu pour créer un nouveau profil vide.
On se place dans l'onglet ETUDE.

5.2 Sélection des fichiers

5.2.1 Définition d'un chemin de base

Dans l'onglet ETUDE, on choisit un chemin de base pour simplifier l'accès aux fichiers.
On clique sur l'icône , on choisit le répertoire `/home/tutorial`.

5.2.2 Ajout de fichiers existants

On ajoute le fichier de commandes en cliquant sur , la sélection de fichier s'ouvre directement dans le chemin de base que l'on vient de définir. Il ne reste qu'à sélectionner le fichier `demo001a.comm` (double-clic ou simple clic + ok), et le fichier apparaît dans la liste. Notons qu'astk identifie le type de ce fichier à partir de son extension « comm », le numéro d'unité logique est positionné à 1, la case « D » (donnée) est cochée.

On fait de même pour le fichier de maillage au format Gmsh (`demo001a.msh`). astk reconnaît l'extension « msh », le numéro d'unité logique est positionné à 19, la case « D » est cochée.
De même pour le maillage du contour `demo001a.18`, astk identifie le type « libr » et positionne le numéro d'unité logique à 18. Il faut cocher la case « D » car il s'agit d'un fichier en donnée.

On peut également ajouter le fichier `demo001a.datg`. On décoche la case « D », il ne sera pas utilisé dans l'étude mais on peut visualiser ce maillage en l'ouvrant avec Gmsh (voir §5.5).

5.2.3 Ajout de fichiers…

Sauf si une exécution a déjà eu lieu, les fichiers résultats n'existent pas encore, on ne peut donc pas les ajouter en parcourant l'arborescence.

5.2.3.1 …en insérant une ligne vide

Le maillage au format Gmsh sera relu et converti dans le fichier de commandes *Aster* par la commande `PRE_GMSH` en maillage au format *Aster*. On peut récupérer ce maillage en ajoutant un fichier de type « mail » sur l'unité logique 20.

On clique sur , une ligne est ajoutée dans la liste. On choisit le type « mail » dans la liste (ce qui a pour effet de positionner le numéro d'unité logique à 20). On indique le nom `/home/tutorial/demo001a.mail` ou `demo001a.mail` ou `./demo001a.mail` (puisque l'on peut indiquer le nom du fichier en relatif par rapport au chemin de base). Le fichier est produit par l'exécution, on coche donc la case « R » (résultat) et on décoche « D ».

Remarque

La commande `PRE_GMSH` utilise par défaut les numéros 19 et 20 avec les entrées/sorties, si l'on modifie le fichier de commande pour relire ou écrire les fichiers de maillage sur d'autres unités, il faut être cohérent avec les numéros indiqués dans astk.

5.2.3.2 …avec « Valeur par défaut »

On pourrait continuer ainsi pour ajouter les autres fichiers, mais on va utiliser la fonction « Valeur par défaut » pour les fichiers suivants. Cette fonction utilise le nom du profil astk pour construire les valeurs par défaut (voir [*§ 2.2.1]/Menu contextuel*), on va donc enregistrer le profil.
On choisit *Enregistrer sous…* dans le menu *Fichier*, on va avec le navigateur dans le répertoire `/home/tutorial`, et dans la ligne *Sélection*, on tape `demo001a` (l'extension `.astk` est automatiquement ajoutée).
Notons que le titre de la fenêtre principale d'astk donne le nom du profil courant. Le titre est maintenant : `ASTK version 1.8.0 - demo001a.astk - /home/tutorial`

On insère une ligne vide en cliquant sur , on choisit le type de fichier « pos », puis on clique avec le bouton droit dans la case du nom de fichier et on choisit « Valeur par défaut » : astk construit un nom de fichier à partir du chemin de base (voir [§5.2.1]), du nom de profil (en retirant l'extension) et du type « pos », soit `/home/tutorial/demo001a.pos`. On voit ainsi : `./demo001a.pos` (nom relatif au chemin de base).
La case « R » a été cochée, et le numéro d'unité logique fixé à 37. Dans le fichier de commande, on a indiqué :

```
IMPR_RESU(UNITE=51, ...)
```

on modifie donc le numéro d'unité logique en conséquence, il suffit de cliquer sur l'ancienne valeur, de l'effacer et de taper 51. Seuls deux chiffres sont affichés dans cette case, pour éviter les erreurs, astk vérifie que les numéros d'unité logique sont compris entre 1 et 99.

On procède de la même manière pour ajouter `demo001a.png` : sélectionner le type « libr », modifier l'unité logique en 53 et le nom du fichier.

De même, on ajoute un fichier de type « mess » et un de type « resu » de cette manière (laisser les numéros d'unité logique par défaut).

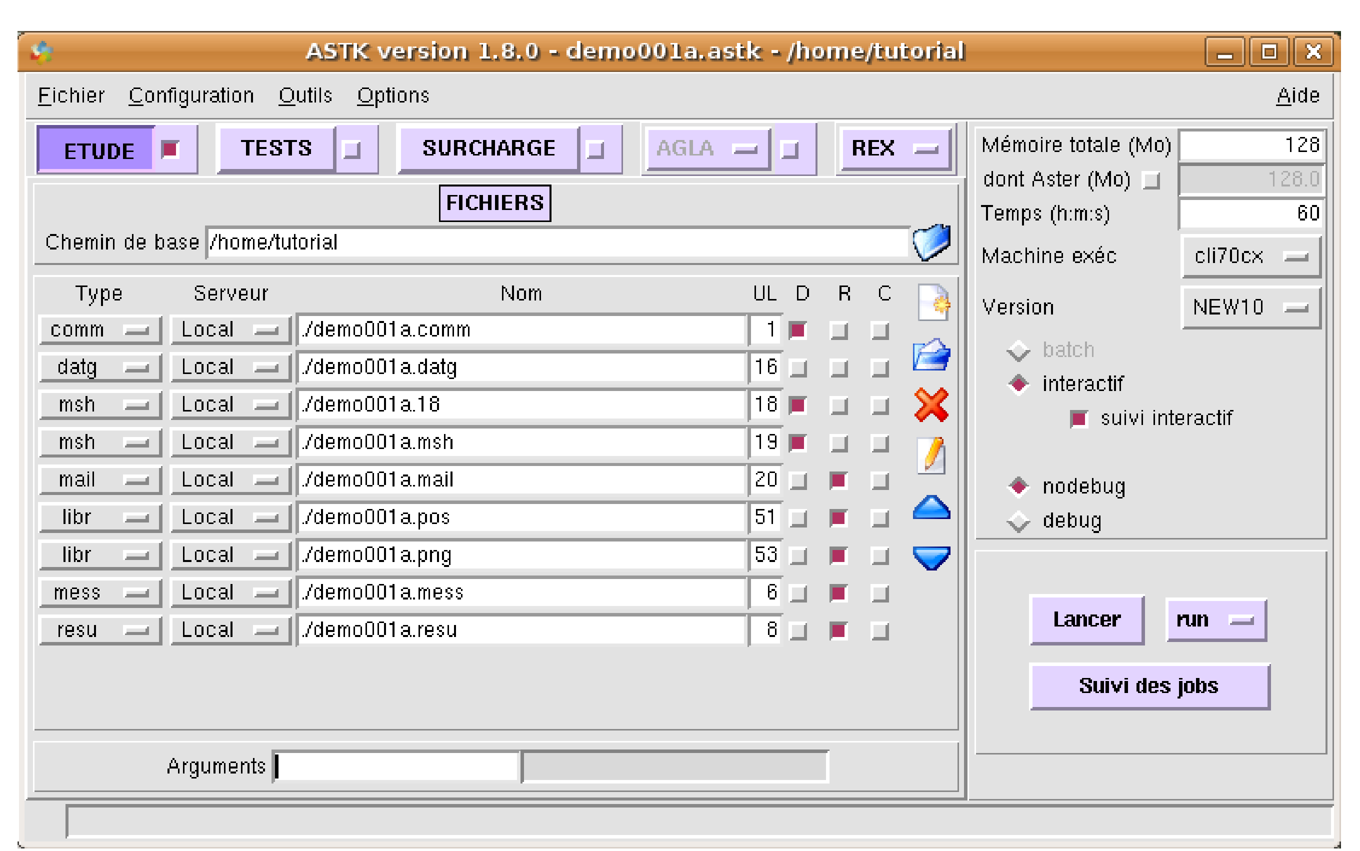


Figure 5.2.3.2-1: Fenêtre du profil d'étude

5.2.4 Supprimer un fichier

Pour supprimer une ligne de la liste des fichiers, il suffit de la sélectionner en cliquant dans la zone où l'on indique le nom du fichier et de cliquer sur l'icône ❌.

Remarque :

> *Seule la référence à ce fichier dans le profil astk est oubliée, le fichier lui-même n'est pas effacé !*

5.3 Lancement du calcul

Les fichiers données et résultats sont sélectionnés, on ajuste les paramètres du calcul (voir [§ 2.3]), et on clique sur le bouton « Lancer ».
On prend soin de cocher la case qui se trouve juste à côté de ETUDE pour signaler que l'on souhaite utiliser le contenu de cet onglet… sinon l'interface nous répond « Rien à lancer ! ».
Si le profil n'a pas encore été enregistré, l'interface demande de choisir un endroit et un nom pour ce profil (voir [§ 5.2.3.2]).

astk appelle as_run pour exécuter le calcul, et transmet au Suivi des jobs (asjob) le numéro du job (numéro du processus en interactif) et d'autres informations qui vont permettre de suivre l'avancement du calcul. L'état initial du calcul est `PEND` (en attente), quand le calcul commence, il devient `RUN`, puis `ENDED` quand il est terminé (d'autres états sont possibles en batch). Le bouton « Actualiser » appelle le service qui rafraîchit l'état des calculs en cours.

Quand le calcul est terminé, on peut consulter l'output du job en double-cliquant sur le job, ou par *Editer/Fichier output*.

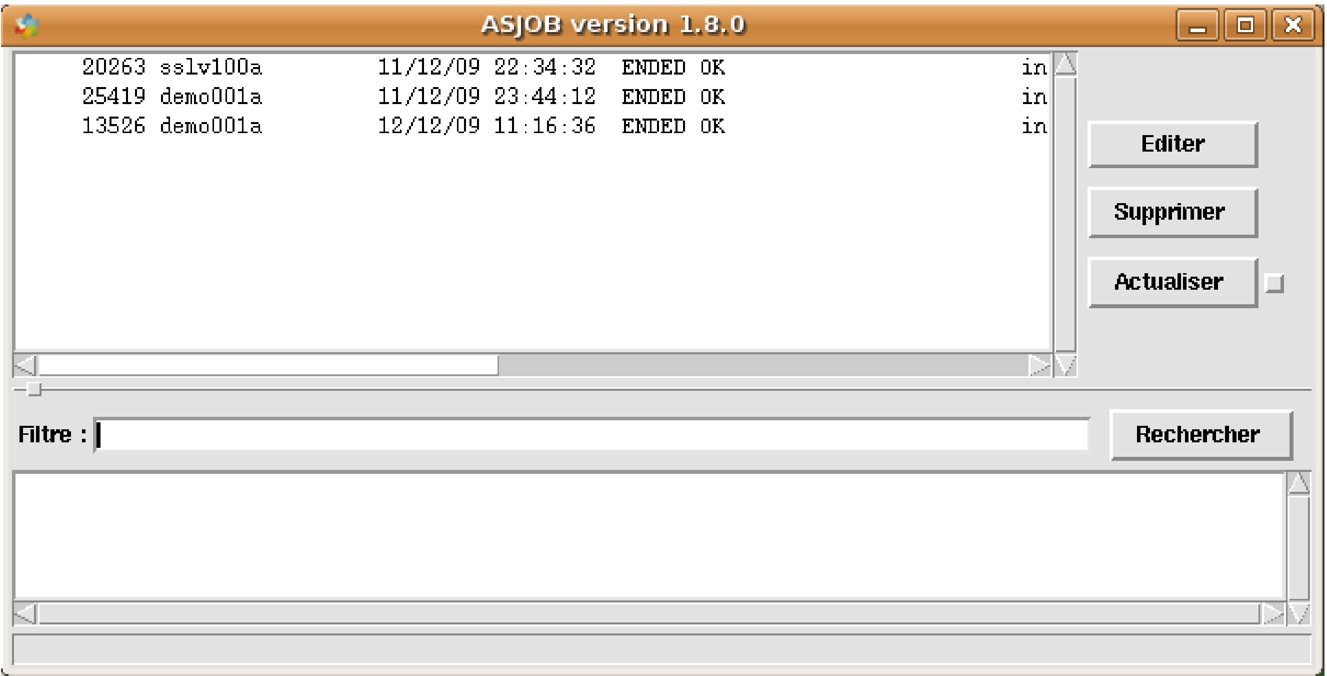


Figure 5.3-1: Fenêtre de suivi des jobs

5.4 Consultation des résultats

On peut consulter les fichiers résultats simplement en double-cliquant sur leur nom, ce qui ouvre un éditeur de texte pour les fichiers « mess » et « resu » ; sur le fichier de résultat au format Gmsh, « pos », cela a pour effet d’ouvrir directement ce fichier dans Gmsh. On visualise ainsi la déformée et les isovaleurs (sous réserve que Gmsh ait été installé, et que « pos » soit dans les types de fichiers associés à Gmsh, voir [*§ 4.3]*).

NB :

- Le répertoire devant accueillir un fichier résultat n'existe pas, il est automatiquement créé si les permissions sont suffisantes.
- Si la copie de fichiers résultats échoue (problème de permissions, de quota, de connexion distante...), ils sont copiés dans un répertoire temporaire sur la machine d'exécution. Une alarme `<A>_COPY_RESULTS` indique le chemin où il faut aller chercher les résultats.

5.5 Utilisation des outils

On peut aussi utiliser astk et le fait que l’on puisse y définir librement des outils pour rassembler dans un profil tous les fichiers nécessaires à une étude même si ceux-ci ne sont pas directement utilisés par *Code_Aster*.

Dans cet exemple, `demo001a.datg` est un fichier que *Code_Aster* ne sait pas relire ; il contient la description de la géométrie, Gmsh l’utilise pour créer le maillage (`.msh`).

On peut néanmoins l’insérer dans le profil (bouton), lui affecter un type quelconque (« libr » par exemple) puisqu’il ne sera pas utilisé lors de l’exécution (cases D, R non cochées).
On peut ouvrir directement la géométrie en faisant *Ouvrir avec…/Gmsh* (clic droit sur le nom du fichier), modifier la géométrie ou les paramètres du maillage, remailler et enregistrer le maillage.

On peut ensuite relancer le calcul sur le nouveau fichier `demo001a.msh`.

Bien évidemment, ceci n’est pas limité à Gmsh ; on peut utiliser d’autres outils (mailleurs, outil de post-traitement, traceur de courbes…) directement depuis astk et accéder ainsi à tous les fichiers d’une étude depuis un profil avec l’outil adéquat.

5.6 Fonctionnalités avancées

5.6.1 exectool

On choisit dans astk la version de *Code_Aster*, le mode debug ou nodebug, une éventuelle surcharge et ceci conduira à utiliser tel ou tel exécutable de *Code_Aster*.
On peut aller encore plus loin en précisant la manière exacte de lancer cet exécutable : c'est le rôle de l'option de lancement **exectool** (menu Options).

En temps ordinaire, *Code_Aster* est lancé avec une commande du type :

```
./aster.exe argument1 argument2 ...
```

Utiliser l'option exectool revient à lancer :

```
cmde_exec ./aster.exe argument1 argument2 ...
```

Dans le menu Options, on peut préciser directement `cmde_exec` ou bien un nom d'outil défini dans le fichier de configuration d'as_run : `[ASTER_ROOT]/etc/codeaster/asrun`

Exemple 1 :
Dans le menu Options, on saisit dans la case exectool : `time`
La commande de lancement sera donc `time aster.exe arguments...`
La commande `time` accepte exactement ce type d'argument (un exécutable et ses arguments), on aura donc le temps d'exécution du calcul. Cela n'a pas grand intérêt, *Code_Aster* donne déjà ce type d'information.

Exemple 2 :
Dans le fichier de configuration `[ASTER_ROOT]/etc/codeaster/asrun`, on définit (sur une seule ligne) :

```
memcheck : valgrind --tool=memcheck --error-limit=no --leak-check=full --suppressions=/opt/aster/valgrind-python.supp
```

Il suffit ensuite d'indiquer dans le menu Options, exectool : `memcheck`
`memcheck` est défini dans le fichier de configuration (à gauche des « : »), donc c'est la commande complète `valgrind...` qui sera utilisée lors du lancement.

On peut définir autant d'outils que l'on souhaite à condition de ne pas entrer en conflit avec les autres paramètres définis dans ce fichier. Pour cette raison, la définition des « exectools » devrait évoluée à l'avenir.

6 Comment réaliser une surcharge ?

On considère dans ce paragraphe que l'utilisateur est familier avec la manipulation des listes de fichiers dans l'onglet ETUDE.

Une surcharge consiste à ajouter ou modifier une partie des sources de *Code_Aster* et à l'utiliser pour réaliser une étude. L'objectif d'une surcharge est de produire un exécutable, un catalogue de commandes compilé et/ou un catalogue d'éléments. On peut aussi surcharger des modules python ;, dans ce cas, les fichiers sources sont recopiés dans le répertoire d'exécution (il n'y a pas d'objet réceptacle des fichiers python surchargés).
On se place dans l'onglet SURCHARGE.

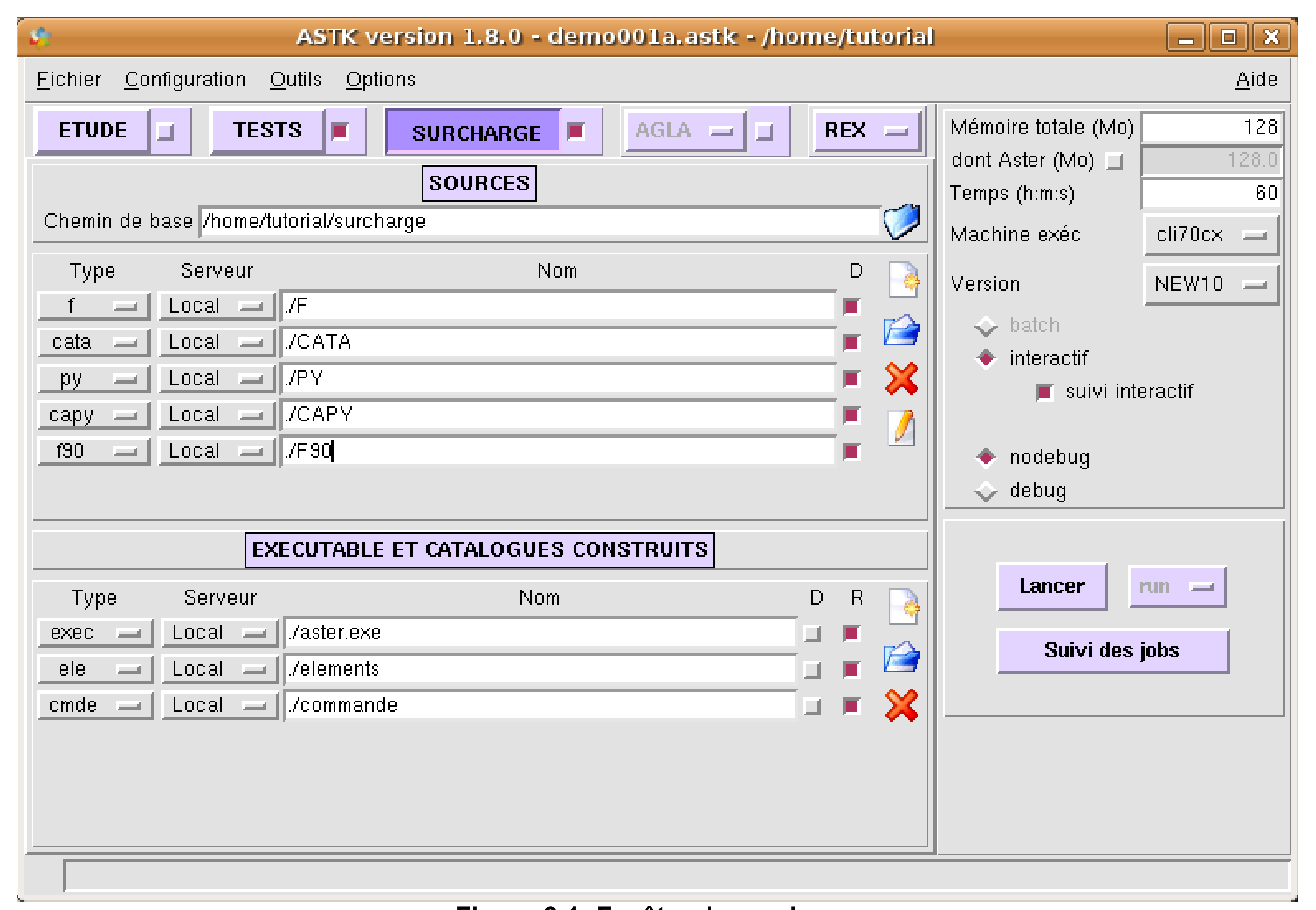


Figure 6-1: Fenêtre de surcharge

6.1 Ajout des sources

On procède comme pour ajouter des fichiers pour une ETUDE. On peut sélectionner soit un fichier, soit un répertoire. Il est souvent plus clair et plus pratique de placer ses fichiers sources dans des répertoires.
Si on ajoute un répertoire dans la liste des sources (partie supérieure de l'onglet SURCHARGE) de type « f » (fichiers fortran), tous les fichiers dont l'extension est `.f` seront compilés et utilisés pour faire un nouvel exécutable.

6.2 Définir les résultats de la surcharge

Les fichiers C (type « c »), fortran (type « f ») et fortran90 (type « f90 ») permettent de construire un exécutable : type « exec ».
Les catalogues de commandes « capy » permettent de construire un catalogue de commandes compilé : type « cmde » (répertoire contenant les fichiers `cata.py` et `cata.pyc`).

Les catalogues d'éléments, d'options et de grandeurs « cata » servent à produire un catalogue d'élements compilés : « ele ».

6.3 Prise en compte de la surcharge

Pour que les données renseignées dans l'onglet SURCHARGE soient prises en compte, il faut cocher la case située juste à droite du bouton SURCHARGE (l'onglet SURCHARGE est toujours coché dans ce paragraphe).

Il faut absolument un réceptacle en résultat correspondant aux sources en données (« D » coché).

S'il y a des répertoires « c » et/ou « f/f90 » en « D »onnée , il faut un « exec » en « R »ésultat ; de même pour « capy » avec « cmde » et « cata » avec « ele ».
astk aide normalement dans cette tâche en cochant/décochant automatiquement les sources nécessaires à la création de l'exécutable ou des catalogues quand on coche/décoche la case « R »ésultat de ceux-ci. Revers de la médaille, on doit décocher certaines lignes quand on jongle avec plusieurs répertoires de sources qui ne doivent pas être compilés ensemble...

On peut préparer la surcharge indépendamment de l'étude (conseillé) ou bien faire la surcharge et lancer l'étude dans la foulée.

Pour préparer la surcharge :
- décocher l'onglet ETUDE
- mettre les réceptacles correspondants en « R »ésultat seulement
- mettre les sources en « D »onnée (astk l'a fait automatiquement)
- « Lancer ».

La surcharge construit les résultats (exécutable, catalogues…) à partir des données (les sources).

Lancement de l'étude en utilisant cette surcharge :
- cocher l'onglet ETUDE
- mettre les réceptables en « D »onnée seulement
- décocher l'indicateur « D » pour les sources (fait automatiquement par astk)
- « Lancer ».

Les produits de la première étape sont alors utilisés comme données pour lancer l'étude.
Attention : Ne pas décocher SURCHARGE, sinon l'étude sera lancée avec la version standard non surchargée.

7 Comment lancer une liste de tests ?

Il est nécessaire d'aborder ce point après les deux précédents. En effet, lancer une liste de tests n'a d'intérêt que pour valider une surcharge soit vis à vis de la non régression des fonctionnalités originales du code, soit sur un ensemble de cas testant une nouvelle fonctionnalité.

Pour utiliser cette possibilité, il faut cocher la case située juste à droite du bouton TESTS (ce qui décoche automatiquement ETUDE). Dans la plupart des cas, on utilise une version surchargée, donc dans ce cas l'onglet SURCHARGE est également coché.
Les tests sont lancés en parallèle en fonction des ressources disponibles (voir §9.1).

Remarque n°1

Il faut absolument préparer la surcharge d'abord, puis lancer les tests avec les résultats de cette surcharge.

Remarque n°2

Sur le serveur de référence, le lancement d'une liste de tests doit être fait uniquement en batch.

Les données sont très simples :
- « list » : un fichier donnant la liste des tests à lancer (obligatoire).

- « rep_test » : un répertoire contenant les fichiers nécessaires au lancement des tests (facultatif). Il s'agit des données des cas-tests dont les noms doivent correspondre à ceux de la liste. Ce répertoire vient toujours en surcharge du répertoire `astest` de la version utilisée.
- « resu_test » : le répertoire où seront écrits les résultats des tests. La copie des résultats est faite selon les paramètres optionnels définis dans le menu Options (voir §2.1.4).

Les traces d'exécution de chaque test sont conservées dans le sous-répertoire `flash` de « resu_test » ou bien dans le répertoire indiqué par l'option « flashdir » du menu Options. Ou bien, encore, ces traces ne sont pas conservées si « flashdir » vaut None.

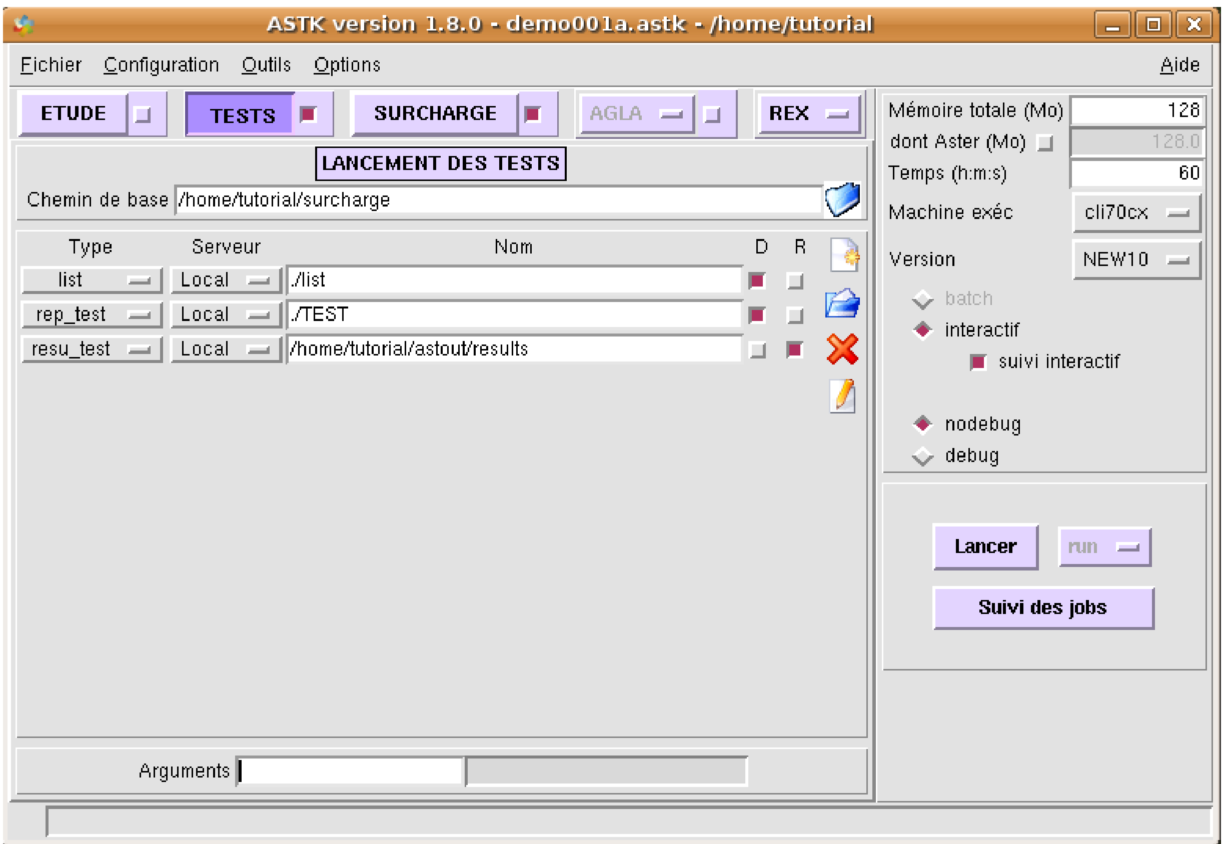


Figure 7-1: Passage d'une liste de cas-tests

8 Comment lancer une étude paramétrique

On entend par étude paramétrique une étude standard (définie dans l'onglet ETUDE) dans laquelle on souhaite faire varier un ou plusieurs paramètres.

L'étude est aussi générale que n'importe quelle étude définie dans astk : on peut donc faire appel à une surcharge si besoin.
Attention dans ce cas, à bien produire tous les résultats avant de lancer l'étude paramétrique et de mettre les résultats de cette surcharge(exécutable, catalogues, sources python) sur la machine d'exécution pour ne pas à transférer les fichiers depuis une machine distante pour chaque calcul.

Remarque

L'étude doit être valide avant d'être déclinée sur le jeu de paramètres. Elle doit donc tourner sans erreur. Il est aussi important d'optimiser l'étude avant de la décliner sur un grand nombre de valeurs des paramètres.

8.1 Définition du jeu de paramètres

La liste des paramètres, et ainsi les N déclinaisons de l'étude nominale, est définie dans un fichier de type « distr ».
Ce fichier doit être à la syntaxe Python, et donc définir la liste des paramètres sous le nom VALE (en majuscules). Chaque item de cette liste définit le dictionnaire des paramètres utilisé pour un calcul. VALE est donc une liste de N dictionnaires.
Dans ce fichier, on peut utiliser `_F` comme dans un fichier de commande pour définir les dictionnaires.

8.1.1 Exemple de fichier « distr »

```
VALE=(
  _F(P1=1.,   P2=0.3),
  _F(P1=22.,  P2=0.4),
  _F(P1=333., P2=0.4),)
```

Ici, on devra donc lancer 3 calculs. Dans le premier calcul, `P1` vaudra `1.`, `P2` vaudra `0.3`. Dans le deuxième, `P1=22.` et `P2=0.4`, etc.

On peut faire appel à de la programmation pour calculer VALE.
Le contexte d'exécution du fichier « distr » est vide (ne contient que `_F`). Il faut importer les modules dont vous avez besoin.

8.1.2 Pré/post-traitements communs à tous les calculs

On dispose de 4 mots-clés dans le fichier « distr » : `PRE_CALCUL`, `UNITE_PRE_CALCUL`, `POST_CALCUL`, `UNITE_POST_CALCUL`.
`PRE_CALCUL` (resp. `POST_CALCUL`) définit un texte (un ensemble de commandes Aster) qui sera inclus juste après `DEBUT` (resp. juste avant `FIN`).
`UNITE_PRE_CALCUL` (resp. `UNITE_POST_CALCUL`) propose le même fonctionnement sauf qu'on fournit un numéro d'unité logique.

Cette modification est faite pour tous les fichiers de type « comm » présents dans le profil.

Cette possibilité est notamment utilisée par la commande de recalage `MACR_RECAL` pour ajouter un post-traitement à chaque calcul esclave.

8.1.3 Utilisation des paramètres dans le fichier de commandes

La déclinaison de l'étude nominale se fait par remplacement des paramètres dans le ou les fichiers de commandes. Le remplacement est fait par expression régulière (l'expression utilisée est `'^ *para *=.*$'`) où `para` est le nom du paramètres (`P1` ou `P2` dans l'exemple ci-dessus).
Cela signifie que la ligne commence par des espaces, puis le nom du paramètre suivi éventuellement d'espace, puis le signe « = ». Tout ce qui suit est remplacé par la valeur du paramètre.

Exemples :

On définit la valeur de P1, puis on utilise P1 dans la suite du jeu de commandes :

```
DEBUT()
P1 = 0
mat = DEFI_MATERIAU(ELAS=_F(E=P1,
                            NU=0.3,))
```

Nom du paramètre mal choisi :

```
DEBUT()
E = 2e11
mat = DEFI_MATERIAU(ELAS=_F(
                            E=E,
```

```
NU=0.3,))
```

Le jeu de commandes produit sera invalide car l'avant-dernière ligne sera substituée par `E=valeur` y compris la virgule !

En cas de doute, il faut vérifier le jeu de commandes après substitution. On le trouve dans le répertoire des résultats (voir § suivant).

8.2 Gestion des résultats

Le répertoire des résultats est défini dans le profil d'étude sous le type « repe » (en « R »ésultat).
On trouvera dans ce répertoire, autant de répertoires que de calculs lancés, nommés automatiquement `calc_i` (par exemple de `calc_01` à `calc_50`). Ceux-ci contiennent le ou les fichiers de commandes après substitution des valeurs du ou des paramètres (nommés `command_j.comm`).
C'est également dans le répertoire `calc_i` que seront copiés les résultat du calcul n°`i`. Pour ne pas être limité sur les résultats à rapatrier, on utilise le mécanisme `REPE_OUT`.

Il est possible de récupérer la base de chaque calcul (base classique, type « base » ou base au format hdf, type « bhdf »). Pour cela, il faut ajouter une entrée dans le profil de type « base » ou « bhdf » dont le chemin **doit être identique** au chemin du « repe ».
On trouvera ensuite la base dans chaque répertoire `calc_i/base` ou `calc_i/bhdf`.
Il est possible ensuite de lancer une poursuite distribuée en mettant simplement la base en donnée. Il est bien évidemment nécessaire que la liste des paramètres utilisés pour la poursuite soit identique à celle du calcul précédent pour que chaque poursuite élémentaire ait la base correspondant au calcul élémentaire initial associé.

Comme pour le lancement de tests, les traces d'exécution de chaque calcul sont conservées dans le sous-répertoire `flash` de «repe» ou bien dans le répertoire indiqué par l'option « flashdir » du menu Options. Ou bien, encore, ces traces ne sont pas conservées si « flashdir » vaut None.

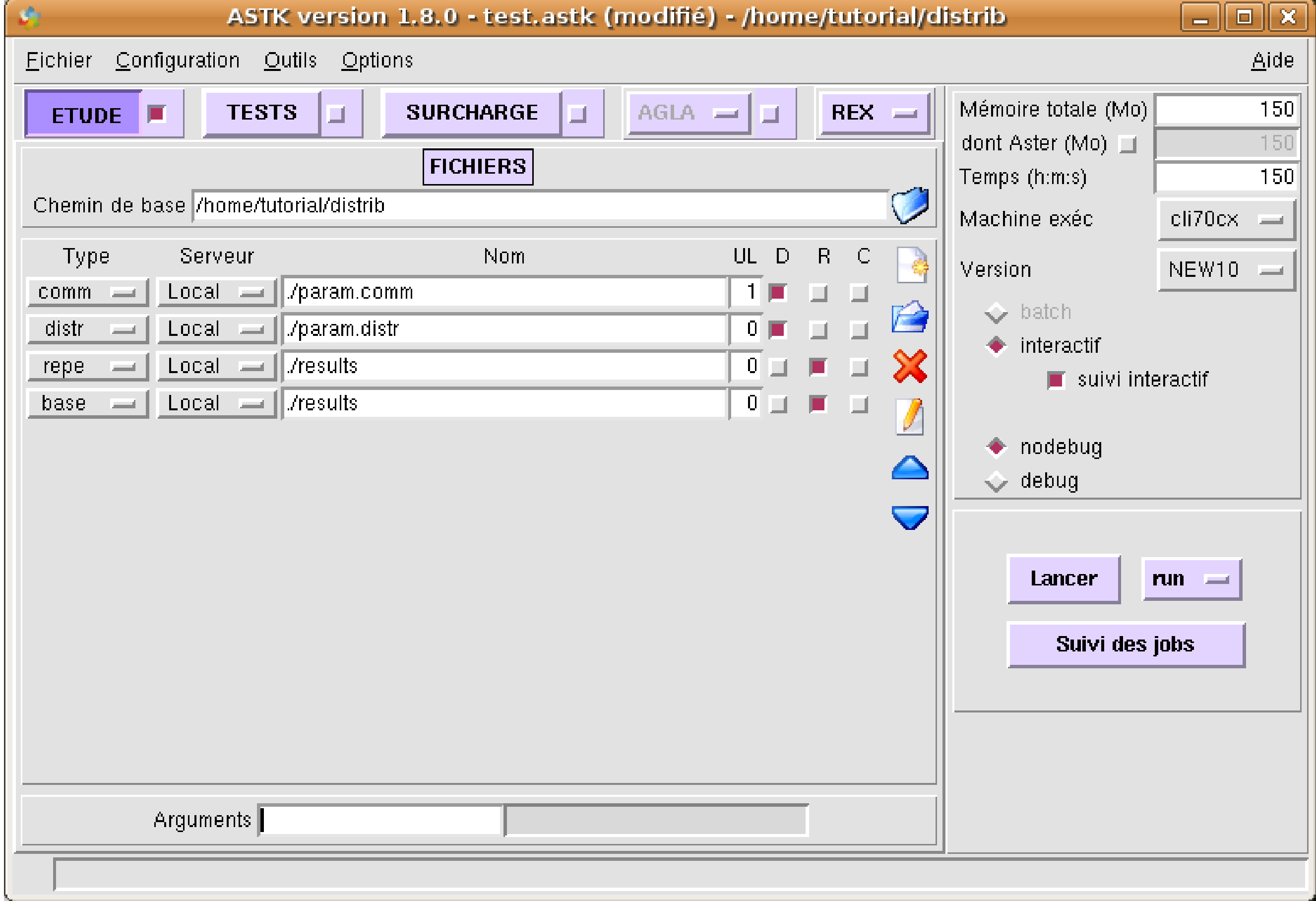


Figure 8.2-1: Profil pour une étude paramétrique

8.3 Lancement

Le lancement est strictement identique au lancement de l'étude nominale seule. Seule l'option « distrib » doit être mise à **oui** dans le menu Options.
Les calculs sont lancés en parallèle : voir §9.1 concernant les options spécifiques.

9 Lancement de calculs en parallèle

9.1 Distribution de calculs

La gestion des calculs distribués est activée lors du lancement d'une étude paramétrique ou de cas-tests. En effet, dans les deux cas, chaque calcul est indépendant des autres. On peut ainsi les soumettre en parallèle pour réduire le temps de retour.

9.1.1 Utilisation des ressources disponibles

On peut insérer dans le profil (onglet ETUDE ou TESTS), un fichier de type « hostfile ». On y définit la liste des machines disponibles et pour chacune le nombre de processeurs et la quantité de mémoire (en Mo) utilisables.

Exemple :

```
[compute01]
cpu=4
mem=8192

[compute02]
cpu=1
mem=1024
```

Cela signifie que jusqu'à 4 calculs pourront être soumis sur `compute01` (dans la mesure où ils ne demandent pas plus de 8192 Mo à eux 4) et 1 calcul sur `compute02` utilisant moins de 1024 Mo.
En batch, on peut soumettre beaucoup plus de calculs que de processeurs disponibles et laisser le gestionnaire de batch répartir les calculs sur un cluster par exemple. Dans ce cas, on peut fixer `cpu=50` pour laisser au maximum 50 calculs dans le gestionnaire de batch.

S'il n'y a pas de fichier « hostfile » dans le profil, on prend celui dont le nom est fixé dans le fichier de configuration `[ASTER_ROOT]/etc/codeaster/asrun` sous le label `interactif_distrib_hostfile` ou `batch_distrib_hostfile` selon le mode de lancement.
Si aucun fichier « hostfile » est spécifié, le nombre de processeurs (coeurs en fait) et la mémoire totale sont automatiquement déterminés.

Remarques

- *On peut facilement écrouler une machine en lançant trop de calculs vis à vis des ressources disponibles. Il est conseillé de se renseigner sur les possibilités d'utilisation de moyens de calculs partagés (classe batch dédiée par exemple).*
- *Les calculs parallèles comptent bien pour le nombre de processeurs qu'ils utilisent et non pas pour 1.*
- *Avant le lancement des calculs, la connexion aux noeuds de calcul est testée. La liste sera limitée aux machines ayant été contactées avec succès.*

9.1.2 Délai d'expiration

Lorsque le nombre de calculs à lancer est très supérieur au nombre de processeurs globalement disponibles (et c'est souvent le cas), des calculs sont en attente de soumission.
Si un calcul demande plus de mémoire qu'aucune machine ne peut en offrir, il resterait indéfiniment en attente.

Pour éviter cela, un délai d'expiration (timeout) est défini égal au temps du calcul maître, c'est-à-dire le temps choisi dans astk lors de la soumission globale.

Si aucun calcul n'a été soumis durant ce délai, le calcul est rejeté.

9.2 Activation du parallélisme de *Code_Aster*

Le parallélisme interne à *Code_Aster* est présent sous deux formes :
- Le parallélisme OpenMP fonctionne en mémoire partagée et est disponible uniquement dans le solveur `MULT_FRONT`. Il faut bien sûr que la version de *Code_Aster* ait été compilée avec les options adéquates.
- Le parallélisme MPI (par envoi de message, Message Passing Interface) est disponible dans le solveur `MUMPS` et dans les calculs élémentaires. La compilation est beaucoup plus compliquée et n'est pas automatique lors de l'installation de *Code_Aster* (il faut choisir une implémentation MPI, compiler les pré-requis et notamment `MUMPS` en MPI, puis *Code_Aster*).

On choisit le nombre de processeurs utilisés en OpenMP et le nombre de processeurs utilisés en MPI (répartis sur un certain nombre de noeuds de calculs) dans le menu Options (voir §2.1.4).

10 Utilisation de as_run

Lorsque l'on utilise l'interface astk pour lancer des calculs, celle-ci en tant que client fait appel à des services proposés par un serveur qui peut se trouver sur la même machine ou une machine distante (dans le cas où le serveur est distant, il y a des échanges de fichiers et une commande shell à travers le réseau (protocole rsh ou ssh) que nous ne détaillons pas ici).

Les fonctionnalités de `as_run` peuvent être classées en plusieurs catégories :

- **pour l'utilisateur** : fonctionnalités qui peuvent être utilisées par les **utilisateurs**, comme par exemple lancer *Code_Aster* « à la main »,
- **pour le développeur** : fonctionnalités utilisées par les développeurs pour visualiser un fichier, copier un fichier, construire une liste de tests, vérifier le catalogue de messages...
- **pour maintenir une version de développement** : construit une version, la mettre à jour...
- **réservées aux clients** : fonctionnalités sans intérêt direct hors de astk (ou un autre client),
- **pour les tâches d'administration** : lien vers l'outil de suivi des anomalies, la base de données d'études...

Les différentes fonctionnalités de `as_run` sont données en tapant : `as_run --help`.

10.1 Pour l'utilisateur

- Lance l'exécution décrite par le profil (action par défaut) :
```
as_run --run [options] user@mach:/nom_profil.export
```

Le fichier export peut être en local ou sur une machine distante.
Chaque ligne du fichier commence par :
- P : définition d'un paramètre,
- A : définition d'un argument de la ligne de commandes de *Code_Aster,*
- F : définition d'un fichier,
- R : définition d'un répertoire,
- N : utilisé uniquement pour l'atelier de génie logiciel (AGLA).

Le format pour les paramètres et arguments est : `P nom_parametre valeur` ou `A nom_argument valeur`.

Pour les fichiers et répertoires, le format est :

```
F/R type chemin DRC unite_logique
```

où `DRC` précise si le fichier ou répertoire est en `D`onnée, `R`ésultat (les deux sont possibles ensemble), et si le contenu est `C`ompressé.

- Exécute rapidement un calcul à partir des fichiers en arguments :

```
as_run --quick [options] file1 [file2 [...]]
```

Les développeurs peuvent indiquer une surcharge de sources fortran (resp. python) avec les options `--surch_fort` (resp. `--surch_pyt`).

- Produit une bibliothèque dynamique nommée `FILE` en compilant les fichiers source `src1`... `srcN`. Utilisé typiquement pour construire une bibliothèque UMAT. :

```
as_run --make_shared --output=FILE [src1 [...]] srcN
```

C'est le compilateur et les options de compilations définies dans le fichier `config.txt`, de la version par défaut ou choisie avec l'option `--vers`, qui sont utilisés pour construire la bibliothèque. `FILE` est le nom de la bibliothèque produite qui sera indiqué dans les mots-clés de *Code_Aster*.

10.2 Pour le développeur

- Affiche un fichier source : fortran, c, python, capy, cata, histor ou test :

```
as_run --show [options] obj1 [obj2...]
```

- Copie un fichier source dans le répertoire courant :

```
as_run --get [options] obj1 [obj2...]
```

- Affiche le diff d'un fichier source : fortran, c, python, capy, cata, histor ou test :

```
as_run --diff [options] obj1 [obj2...]
```

- Retourne la subroutine principale d'une commande code_aster :

```
as_run --getop [options] commande[.capy]
```

- Construit un fichier export pour lancer un cas-test et l'imprime sur stdout :

```
as_run --get_export testcase_name
```

- Construit une liste de cas-tests à partir de commandes/mots-clés et/ou vérifiant des critères de temps cpu ou mémoire :

```
as_run --list [--all] [--test_list=FILE] [--filter=...]
   [--command=...] [--user_filter=...] [test1 [test2 ..]]
```

- Construit le diagnostic des cas-tests code_aster (depuis dir ou le répertoire astest par défaut) et écrit un fichier pickled du résultat :

```
as_run --diag [--astest_dir=DIR1,[DIR2]] [--test_list=LIST]
   [--only_nook] [diag_result.pick]
```

- Opération sur les catalogues de messages de code_aster. subroutine = retourne les messages appelés par "subroutine". message_number = retourne les routines utilisant ce message. check = vérifie les catalogues et affiche quelques statistiques. move = déplace un message d'un catalogue à un autre et produit les catalogues et fichiers source modifiés. :

```
as_run --messages subroutine | message_number
as_run --messages check [--fort=...] [--python=...] [--unigest=...]
as_run --messages move old_msgid new_msgid
```

- Retourne les informations sur les processeurs et la mémoire des machines données :

```
as_run --get_infos [--output=FILE] host1 [host2 [...]]
```

Remarques

Le script d'installation crée des liens symboliques `show`, `get`, `getop` vers as_run dans `[ASTER_ROOT]/bin`, ce qui permet de ne taper que `show` au lieu de `as_run -show`.
De même qu'`astk` peut être invoqué par `codeaster-gui`, `as_run` peut être appelé par `codeaster`.

10.3 Pour maintenir une installation locale

- Retourne le numéro de la version de développement :
```
as_run --getversion [options]
```

- Construit une version de code_aster (exécutable, bibliothèques, catalogues). `target` peut être all ou clean :
```
as_run --make [--vers=VERS] [target]
as_run --make [--vers=VERS] clean
```

Exemple : `as_run --make clean bibc/hdf` supprime les fichiers objets associés aux fichiers sources du répertoire `bibc/hdf`.

- Effectue une ou plusieurs mise à jour d'une version de développement (préférer --auto_update qui se charge de télécharger toutes les mises à jour et de les appliquer dans le bon ordre) :
```
as_run --update [options] fich1.tar.gz [fich2.tar.gz...]
```

- Télécharge les mises à jour disponibles depuis un serveur et les applique à une version de développement :
```
as_run --auto_update [--vers=...] [--force_upgrade]
   [--keep_increment] [--report_to=...]
```

L'option `keep_increment` signale que toutes les versions intermédiaires sont conservées (exécutable, catalogues, modules python).
L'option `force_upgrade` permet de passer par exemple de la version 10.0.30 à la version 10.1.1 (ce qui est licite car la 10.0.30 a été stabilisée sous le numéro 10.1.0).
L'option `report_to` envoie un mail résumant la mise à jour (succès ou échec).

- Télécharge les mises à jour disponibles depuis un serveur et met à jour astk/asrun lui-même :
```
as_run --astk_update [--report_to=...] [version]
```

`version` permet d'installer une version particulière d'astk (valeur du type 1.8.0).
NB : `astk_update` est appelé avant chaque mise à jour par `auto_update`.

- Construction des ctags :
```
as_run --ctags [--vers=VERS]
```

10.4 Pour les interfaces lançant des calculs

- Lance une exécution (en appelant as_run dans un processus séparé) :
```
as_run --serv user@mach:/nom_profil.export
```

- Demande à un serveur de lancer une exécution (en appelant as_run --serv sur le serveur) :
```
as_run --call_server user@mach:/nom_profil.export
```

Cette option est destinée à être appelée sur une machine cliente (directement par l'interface graphique par exemple).

- Envoie le contenu de "filename" (éventuellement sur une machine distante) aux adresses email1, email2,... :

```
as_run --sendmail [--report_to=EMAIL1,EMAIL2] filename
```

- Retourne les informations sur la configuration d'un serveur : batch, interactif (oui/non, limites), noeuds de calcul, versions :

```
as_run --info
```

- Retourne l'état, le diagnostic, le noeud de calcul, le temps cpu et le répertoire de travail d'un calcul :

```
as_run --actu job_number job_name mode
```

- Ouvre le fichier output ou error sur l'écran fourni :

```
as_run --edit job_number job_name mode output|error DISPLAY
```

- Affiche la fin du fichier fort.6 ou les lines correspondant au filtre :

```
as_run --tail job_number job_name mode fdest nb_lines [regexp]
```

- Arrête un calcul et détruit les fichiers associés :

```
as_run --del job_number job_name mode [node] [--signal=...]
```

- Supprime les fichiers des calculs qui ne sont pas dans la liste :

```
as_run --purge_flash job_number1 [job_number2 [...]]]
```

- Convertit un fichier (éventuellement distant) au format html et écrit le résultat dans FILE :
en version bêta

```
as_run --convert_to_html [user@machine:]file --output=FILE
```

10.5 Pour les tâches d'administration

- Insère une nouvelle entrée dans le système de suivi d'anomalies et copie les fichiers joints si un fichier export est fourni :

```
as_run --create_issue issue_file [export_file]
```

- Remplit les champs "corrvdev" ou "corrvexpl" (selon vers) dans les fiches trouvées dans `histor` et éventuellement les ferme :

```
as_run --close_issue --vers=VERS histor
```

- Extrait le contenu des fiches listées dans `input_file` dans `histor` :

```
as_run --extract_histor [--status=STAT] [--format=FORM]
   [--all_msg] input_file histor
```

- Prépare le profil pour insérer une exécution dans la base de données :

```
as_run --insert_in_db [export_file]
```

10.6 Options

```
--version              show program's version number and exit
-h, --help             show this help message and exit
-v, --verbose          rend le programme plus bavard
--silent               fonctionne aussi silencieusement que possible
-g, --debug            affiche les informations de debug
--nodebug_stderr       supprime l'impression des informations de debogage sur
                       stderr
-f, --force            force les opérations qui peuvent utiliser un cache
                       (téléchargement, compilation...)
--display=DISPLAY      valeur de la variable DISPLAY (NB : certaines
                       fonctions lisent cette valeur dans un fichier)
--rcdir=DIR            utilise le répertoire de ressources $HOME/'DIR' (le
                       défaut est .astkrc)
```

```
--remote_shell_protocol=REMOTE_SHELL_PROTOCOL
                      protocole utilisé pour le lancement de commandes
--remote_copy_protocol=REMOTE_COPY_PROTOCOL
                      protocole utilisé pour la copie de fichiers
--editor=EDITOR       ligne de commande de l'éditeur
--devel_server_user=DEVEL_SERVER_USER
                      identifiant sur le serveur de développement
                      (nom/adresse ip généralement défini dans
                      /etc/codeaster/asrun)

Options pour les opérations de maintenance:
  --filter=FILTER     filtres appliqués aux paramètres des tests : 'nom_para
                      < valeur' (comparaisons supportées <, >, =).
  --vers=VERS         Version de Code_Aster à utiliser
  --force_upgrade     Force la mise à jour vers la prochaine version (par
                      exemple de 10.1.xx vers 10.2.0)
  -o FILE, --output=FILE
                      redirige le résultat dans FILE au lieu de stdout
  --surch_pyt=REP     un ou plusieurs répertoires (séparés par une virgule)
                      contenant les fichiers python ajoutés
  --config=FILE       utilise un autre fichier "config.txt" (seulement pour
                      make, update and auto_update).
  --only_nook         rapporte seulement les erreurs (le temps passé dans
                      les tests ok est compté)
  --surch_fort=REP    un ou plusieurs répertoires (séparés par une virgule)
                      contenant les fichiers fortran ajoutés
  -a, --all           copie de tous les fichiers du cas-test
  --destdir=DIR       répertoire racine (fake-root) où les fichiers seront
                      copiés
  --keep_increment    met à jour une version incrément par incrément en
                      conservant les fichiers exécutables intermédiaire
  --search=REGEXP     conserve les tests vérifiant l'expression régulière
                      donnée (ou une simple chaine de caractères).
  --unigest=FILE      un fichier unigest (pour la suppression)
  --command=COMMANDE[/MOTCLEFACT[/MOTCLE[=VALEUR]]]
                      conserve les tests utilisant la commande  et les mots-
                      clés donnés
  --test_list=FILE    liste des cas-tests
  --report_to=EMAIL   adresse de messagerie où envoyer le rapport d'une
                      exécution (seulement pour --auto_update)
  --user_filter=FILE  fichier contenant les classes testlist.FILTRE. Voir
                      [...]/share/codeaster/asrun/examples/user_filter.py
                      comme exemple.
  --astest_dir=DIR    répertoire des cas-tests à analyser
  -l, --local         les fichiers ne sont pas cherchés sur le serveur
                      distant mais sur la machine locale
  --nolocal           force la recherche des fichiers sur le serveur distant
                      (inverse de --local)

Options for operations on jobs:
  --signal=SIGNAL     envoie un signal au calcul (KILL|USR1)
  --result_to_output  Redirige le résultat dans FILE au lieu de stdout

Options pour l'interface au REX:
  --status=STAT       une erreur se produit si les fiches ne sont pas dans
                      cet état
  --all_msg           recupère tous les messages des fiches
  --format=FORM       format de l'histor généré (texte ou html)
```

10.6.1 Précisions supplémentaires

- Quand on récupère un fichier source avec `get`, `show` ou `getop`, celui-ci est mis dans `/tmp/astk_`login`/cache`. Si on demande de nouveau le même fichier, celui-ci est pris directement dans ce répertoire sauf si l'option `--force` est activée ; les fichiers qui ont plus d'une journée sont automatiquement supprimés du cache.

- `--local`/`--nolocal` : ces options sont utilisées quand le comportement par défaut ne convient pas. Dans l'utilisation de `show/get`, il est préférable d'accéder aux sources locaux, c'est donc le défaut. Dans ce cas, on utilisera `--nolocal` pour accéder à une version sur le serveur de référence. Lors de la maintenance d'une installation locale, le défaut est de télécharger les mises à jour sur un serveur distant. Néanmoins, si on récupérer les paquets en local, on peut utiliser `--local` pour les installer.

- Le fichier de configuration principal est `[ASTER_ROOT]/etc/codeaster/asrun`, l'utilisateur peut définir certaines valeurs dans `$HOME/.astkrc/config` (où `$HOME` est le répertoire par défaut de l'utilisateur), en particulier le nom d'utilisateur à utiliser sur le serveur distant (`devel_server_user`) pour éviter une alarme à chaque lancement de `as_run`.

11 Questions Fréquentes

« Rien ne se passe quand on essaie de lancer un calcul, d'éditer un fichier ou d'ouvrir un terminal sur un serveur distant » *ou bien* ***« Code retour = 2, Profile copy failed dans la fenêtre des messages au lancement d'un calcul »***
Il est probable que la machine locale ne puisse pas communiquer correctement avec le serveur distant. Les communications utilisant le protocole rcp/rsh, les fichiers `.rhosts` sur la machine locale et sur les serveurs distants doivent être correctement renseignés. Si on utilise le protocole scp/ssh, il faut que les clés privées et publiques soient cohérentes.
On peut le vérifier en quittant astk, et en le relançant de cette manière :

`astk --debug 0 --check` (`--debug 0` sert à limiter les impressions)

Lire attentivement les informations qui sont imprimées. astk fournit des informations sur la variable d'environnement DISPLAY (vérifier qu'elle est correctement définie), et teste la connexion entre la machine où est lancée astk et les serveurs distants. En cas de problème de connexion avec un serveur, astk suggère une modification du fichier `.rhosts` de la machine cible.

« Quand on soumet un calcul en batch, on a le message : Le numéro du job et la classe n'ont pas pu être récupérés »

Le temps ou la mémoire demandé est probablement au delà des limites des classes de travaux. Voir dans la fenêtre des messages, il y a probablement un message du gestionnaire de batch du style "Cannot exceed queue's hard limit(s)".
Diminuer le temps ou la mémoire, ou choisir explicitement une classe batch qui convient dans les options supplémentaires.

« En cliquant sur Valeur par défaut le nom de fichier reste vide »
Cette fonction utilise le nom du profil pour déterminer une valeur par défaut, il faut enregistrer le profil avant de pouvoir utiliser cette fonction.

« Sur la machine de référence : les comportements en batch et en interactif sont différents »
C'est possible. En batch, *Code_Aster* est lancé par les scripts de l'AGLA, en interactif, on s'appuie sur `as_run`. Signaler dans une anomalie les différences.